QUAECUMQUE VERA

THE UNIVERSITY OF ALBERTA

# A PATTERN RECOGNITION PROGRAM FOR SIMPLE LINE CONFIGURATIONS WITH APPLICATION TO ARABIC NUMERALS AND PRINTED CAPITAL LETTERS

by

Wilma J. Weber McKinney

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

APRIL, 1968

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled A PATTERN RECOGNITION PROGRAM FOR SIMPLE LINE CONFIGURATIONS WITH APPLICATION TO ARABIC NUMERALS AND PRINTED CAPITAL LETTERS submitted by Wilma J. Weber McKinney in partial fulfilment of the requirements for the degree of Master of Science.

## ABSTRACT

An investigation into the problem of machine recognition of simple line patterns was made with an emphasis on the initial configuration from the input device. The main concern was the information coming from the input device and its implications for a pattern recognition program. This resulted in a proposal for an input device structured after the human fovea and in the development of a program which employs a representation for simple line patterns that is believed to lend some insight into mechanical pattern recognition. The program was imbedded in an operational, simulation model of an overall pattern recognition system, which includes a hand simulation of the proposed input device, and found to have a high degree of success.

## ACKNOWLEDGEMENTS

To Professor K.V. Wilson, I express my appreciation and thanks for his guidance in the preparation of this thesis. Also, I wish to thank Professor G.H. Syms for his interest and assistance in this topic.

In addition, to Dr. D.B. Scott, Head of the Department of Computing Science, University of Alberta, my thanks for providing the facilities and assistance for carrying out this research.

TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER I

# INTRODUCTION TO PATTERN RECOGNITION

The term Pattern Recognition may be applied to many areas of investigation. Taxonomists look for cluster patterns with respect to a set of attributes to determine classifications. Mathematicians isolate invariant patterns in logical statements to form theories. Psychologists search for patterns of relationship between stimuli and behavior. Physiologists attempt to determine how the human brain works in recognizing patterns. Computing Scientists use computers to search for patterns and investigate proposed models for pattern recognition. As can be seen from these examples the word "pattern" is used in many contexts.

Pattern is a concept of varied definitions. A dictionary (Barnhart, 1953) gives "an original or model proposed for or deserving of imitation. Anything fashioned or designed to serve as a model or guide for something to be made". Minsky (Feigenbaum and Feldman, 1966) states that "we often use the term teleologically to mean a set of objects which can in some (useful) way be treated alike". Colwell (1965) uses the term pattern to mean the spatial arrangement of objects. He implies that a pattern is the repetition of certain general forms or relationships characteristic of objects. Uhr (1966) states that "patterns seem to be

precisely the wholes that are more than the sum of the parts". Further he implies that patterns are complexes, structures, interactions, grammars and syndromes. Here we shall think of pattern as a structure.

A dictionary (Barnhart, 1953) definition of recognize is "to know again; to perceive to be identical with something previously known; to identify from knowledge of appearance or character". Neisser, 1967, gives "whenever a stimulus evokes a single response consistently, we can say that it is being "recognized"".

We shall define pattern recognition as: to identify a structure from knowledge of its character. This definition implies two phases in pattern recognition. Phase one is to determine the character or the aggregate of qualities that distinguishes one thing from other things; i.e., discover the features or attributes of a structure that can be the criteria for a category. We can think of this phase as learning and we shall refer to it as such. Phase two is to identify a structure from its aggregate of qualities; i.e., find those features of a particular structure that will give it membership in a category. This phase can be thought of as recognition.

In the learning phase in order to distinguish one thing from other things it would be helpful to know, what other things? If the other things are delimited and form a class

certain attributes can be assumed and only intraclass features need be scrutinized. A hierarchy of learning is proposed such that initially broad classes are isolated, according to criteria, from the universe of structures. Names are given to such classes, i.e., alphabet, animals, functions, flowers, etc. The classes serve to focus what needs to be learned. Once a class is determined one proceeds to name members. Each member is essentially a category determined by specific criteria. These categories can in turn be thought of as classes which are likewise subdivided, etc. Consider the class, alphabet. In this class we have the categories denoted by letter names A, B, etc. The A category has a set of distinctive characteristics which are common for many forms of A and separate it from other similar categories, B, C, etc., in the class, alphabet.

Often the learning process is performed unconsciously by human beings. We may not be aware of what is learned that causes a member of a class to become a single category and not a set of categories, or members of a universe to be separated into classes. What do we see in a single alphabetic character presented to us in various forms that causes us to learn it as one name? What criteria do we use for the "A" category; i.e., what characteristics of "A" are vital to its representation? Research has given no definitive answer to this question.

It can be seen that recognition is closely associated with learning. The same process of isolating a class from the universe is performed. Psychological studies of human perception (Rubin and Walls, 1965) have implied that pattern recognition is faster when the class is delimited. Furthermore it was proposed that the smaller the class, i.e., the fewer categories in a class, the faster the recognition of the individual members. When subjects were told a class, say, alphanumeric characters, which includes another class, numeric characters, of which no members, numerals, were present in the set to be recognized the speed of recognition was slower than when the subjects were told only the applicable class, alphabet.

Once the class is determined we proceed to determine a category by extracting the information needed to associate the pattern to be recognized with some pattern we have learned. We do not know what information is used nor how it is used to associate what we see with a learned name.

Thus far we have discussed pattern recognition in general with allusions to a hierarchy of classes leading from recognition of broad classes to succeedingly more specific categories. We have noted that little is known about how humans perceive patterns. Nevertheless there are many attempts to build machines which will duplicate human performance. Some of the many practical uses are

document reading in conjunction with an information storage and retrieval system, direct input of data into the computer, say, from purchasing and sales forms for inventory control, etc.; the storage and subsequent matching of fingerprints, automatic analysis of biomedical pictures, extraction of data from aerial photographs; direct communication with the computer by the spoken word, etc. A useful pattern recognition machine should duplicate or improve upon human performance. It must be economic, accurate and fast.

Patterns occur in visual and other sensory stimuli, in language and other symbols, in assessments and diagonsis and, in general, in descriptions of any complex domain. This thesis is primarily concerned with machine recognition of static, two-dimensional, black and white, visual patterns which are simple configurations of lines. Such line patterns as a two-dimensional perspective drawing of a three-dimensional figure are not considered. The specification lines disallows solid patterns like, say, filled-in squares. Henceforth the word "pattern" will imply these specifications unless otherwise stated. Examples of such patterns are elementary geometrical shapes, printed and written alphanumeric characters, punctuation symbols, etc.

The final goal is to impart pattern recognition capability

to a general purpose digital computer having auxiliary sensing equipment. Our investigation concentrated on the input aspects of the overall problem. The main concern was the information coming from the input device and its implications for a pattern recognition program. This resulted in a proposal for an input device structured after the human fovea and in an hypothesis on how patterns may be represented and subsequently recognized by machine. To implement our proposal and hypothesis, a simulation model of an overall pattern recognition system was developed; this system, the PR system, includes a hand simulation of the proposed input device and the CHAR (mnemonic for character) pattern recognition program which embodies the hypothesis.

An inherent difficulty in pattern recognition is providing a representation for patterns. Several approaches have been used employing various representations. As background and support for our approach to the pattern recognition problem a survey of approaches with reference to specific programs is presented in Chapter II. Several operational programs are discussed in varying detail. Chapter III is the most significant. Here we give the chief implications of our investigation, a description of the proposed input device and coding, and the details of a representation for patterns. Chapter IV contains a

description of the PR (pattern recognition) system with emphasis on the functions of the CHAR program. Chapter V gives our results and conclusions. The mode of simulating input and some samples are in Appendix A. Appendix B contains flow diagrams of the PR system and the CHAR program and listings of the routines. Finally an example of program performance is given in Appendix C.

# CHAPTER II

# SURVEY OF APPROACHES TO THE PATTERN RECOGNITION PROBLEM

## 2.1 Introduction

Research on the general problem of pattern recognition has been for two complementary purposes. One purpose of the research is to discover how human beings extract patterns from their environment, code them for retention and use them to recognize future patterns. The other purpose of the research is to develop computer programs which have a general pattern recognition capability. Investigation for both of these purposes has taken three main approaches which are here called the random approach, the template approach, and the features approach.

## 2.2 The Random Approach

One school of thought concerning the human learning-recognition process proposes a self-organizing system (Hebb, 1961). That is the human brain begins with a random arrangement of neuron connections and organizes itself to categorize the stimulus patterns it receives. Corresponding to this position is a pattern recognition machine which begins with a random, loose organization and organizes itself, thru learning, to recognize a set of patterns

[illegible]

presented to it. In order to build such a machine one must predicate how the machine will change its random organization into an effective organization. Often the researcher formulates his predication from hypothesis of brain functioning. Then if after learning the machine achieves accurate recognition performance we could say it supports the hypothesis. In other cases the researcher formulates his predication without relevance to brain functioning but perhaps using a mathematical or empirical viewpoint. Here the goal is to obtain a recognition machine or to investigate how such a machine might be obtained.

The random machines usually employ either a neural-net model or decision surfaces defined by discriminant functions, or some combination of both. Neural-net models are generally used for simulating the brain. Discriminant functions are generally utilized in pattern recognition programs which depend upon the likelihood of activation in randomly chosen n-tuples of sensing units. Such programs are referred to as trainable pattern classifiers or adaptive decision networks.

A neural-net is composed of neurons modeled from a hypothesis of biological neurons. One such model is the formal neuron of McCulloch and Pitts (see figure 1). In a neural-net there are a number of neurons which are interconnected at random. Some of the connections are

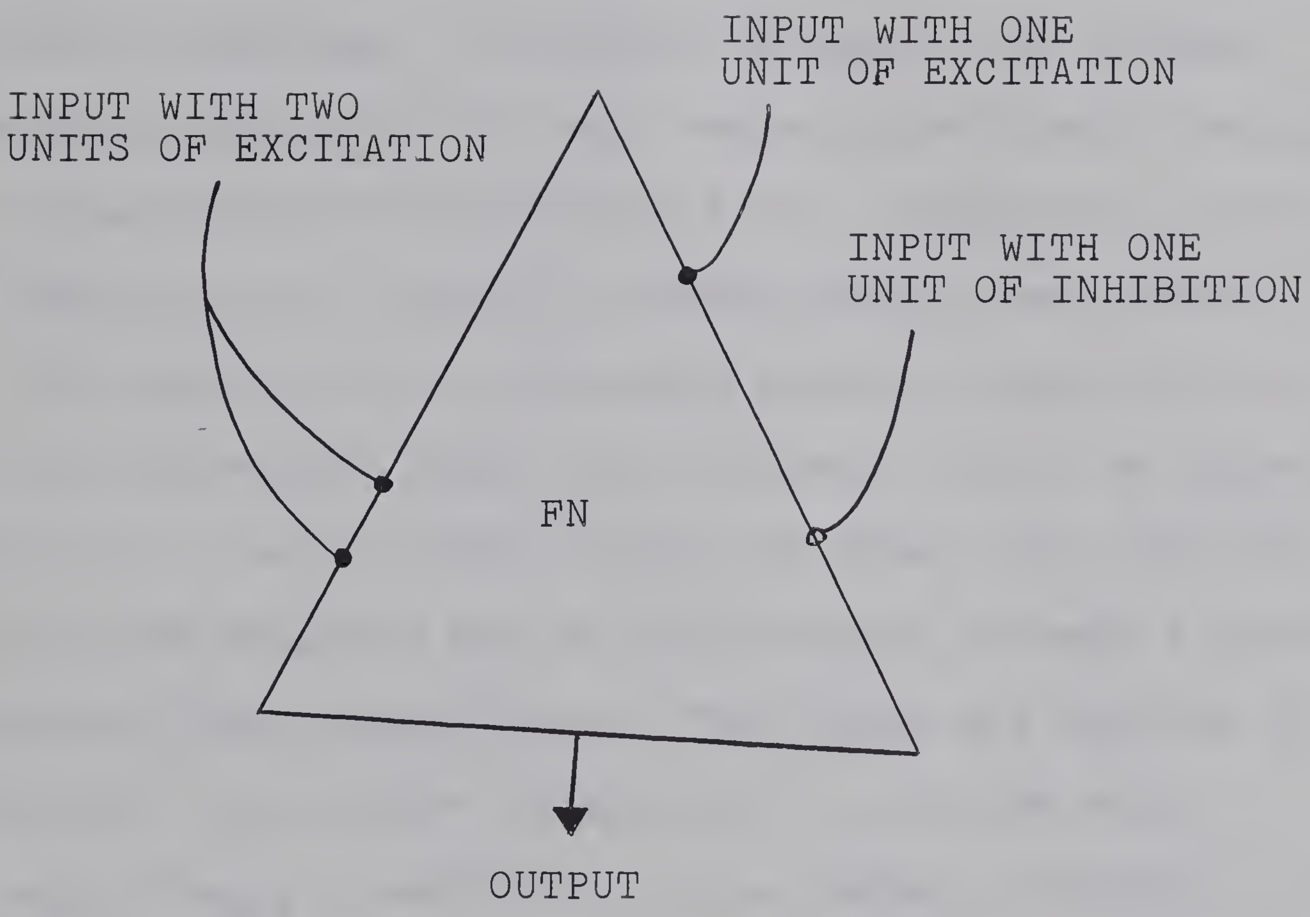

INPUT WITH ONE UNIT OF EXCITATION
INPUT WITH TWO UNITS OF EXCITATION
INPUT WITH ONE UNIT OF INHIBITION
FN
OUTPUT

Figure 1. Symbolic representation of McCulloch and Pitts' formal neuron (Carne, 1965)

inhibitory in nature and some are excitatory. Each neuron has a structure, based upon a neuron model, including a threshold and several efferent and afferent connections with other neurones. The peripheral neurons receive stimuli or elicit response. Repeated stimuli are applied to the model to affect its reorganization. This reorganization is achieved in various ways such as changing thresholds, the nature of connections, or the configuration of interconnections. The model is said to be trained when a specific stimulus will cause a predictable response. The organization will continue to be changeable to allow for adaptation to slightly altered and/or new stimuli.

The basic unit in trainable pattern classifiers is the TLU (threshold logic unit, Nilsson, 1965), as shown in figure 2. A set of input lines are fired into the unit, and, if the weighted sum of these inputs exceeds a specified threshold, the output fires. The inputs are denoted by the variables $x_i$ and the weights by $w_i$ in the sum $\sum_{i=1}^{d} w_i x_i + w_{d+1}$, where $d$ is the number of inputs to the TLU and $w_{d+1}$ is a variable representing the threshold. This sum is a simple discriminant function. Discriminant functions may be non-linear, i.e., quadratic etc., depending upon the relationship between the pattern measurements and the input variables $x_i$. In training, the weights are adjusted to produce the desired outputs. Although the

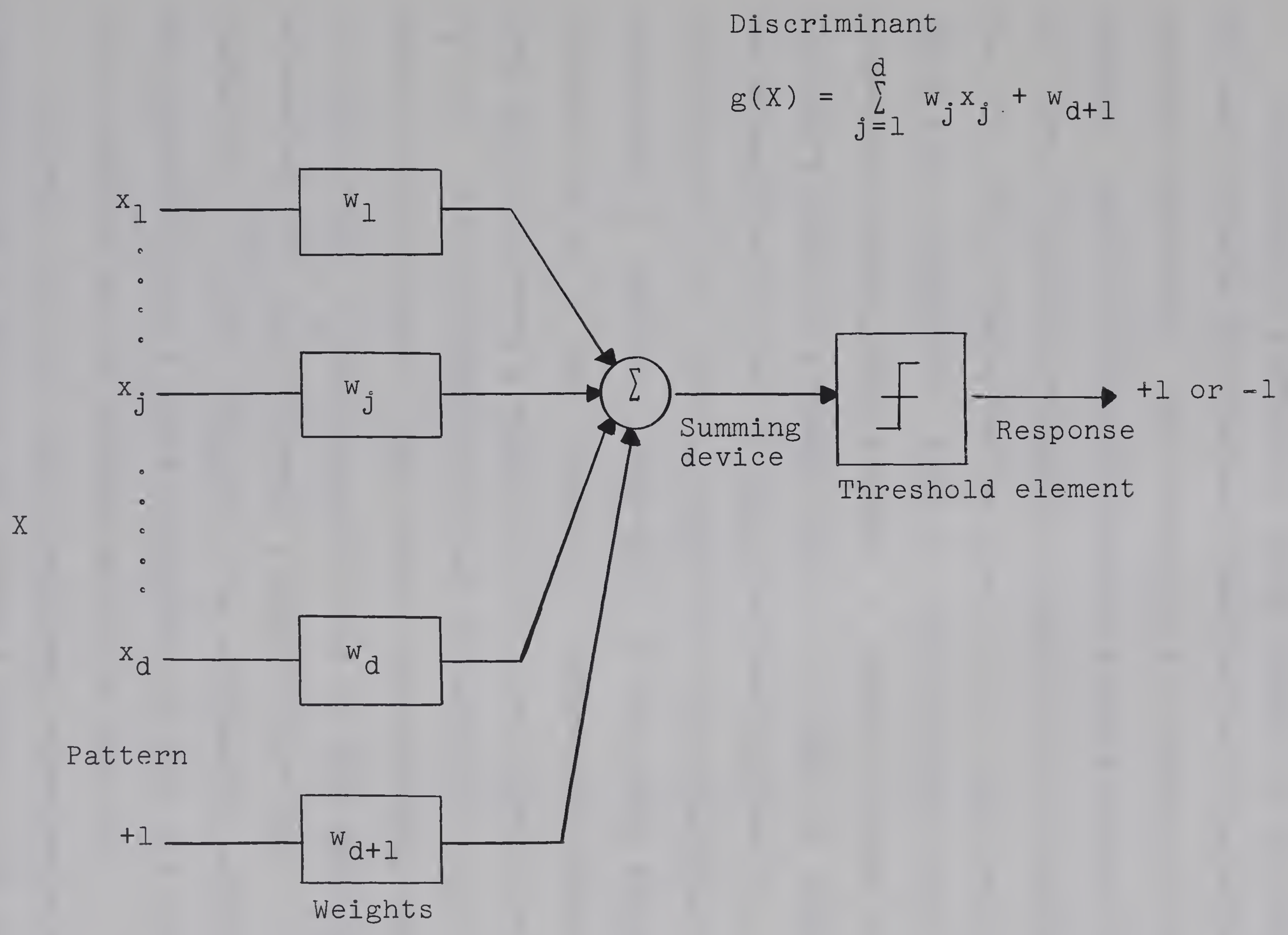


Figure 2. The threshold logic unit (TLU). (Nilsson, 1965).

TLU is similar to a neuron, it is not a specific attempt at duplication of biological structure.

Rosenblatt's Perceptron is an example of the use of a TLU and a simple discriminant function in a trainable pattern classifier. The Perceptron is composed of three layers. Layer one is a rectangular sensing device composed of a set of smaller rectangles congruent to one another, the $S$ units. Layer two is a set of $A$ units each of which has a threshold. Each $S$ unit of layer one is connected randomly with several $A$ units of layer two. The third layer is a single unit, $R$. Each $A$ unit has one connection to the $R$ unit, denoted by some $x_i$. The activation of $S$ units in layer one sends impulses to the connected $A$ units in layer two. If the total impulses received by an $A$ unit, say $A_k$, exceeds the threshold, that unit will send a signal to the third layer, i.e., $x_k = 1$. If the threshold is not exceeded, or $x_k = 0$, there is no response. This occurs simultaneously for all $A$ units. The $R$ unit elicits a one or zero depending on whether $\sum_{i=1}^{d} w_i x_i$ exceeds a threshold or not, $d$ is the number of $A$ units. Figure 3. is a schematic perceptron. The perceptron has a chief limitation in that the location of the pattern on the sensing device is critical.

Roberts, (in Uhr, 1966), performed experiments with a perceptron-like machine having 2048 S units, (a 64 × 64 bit

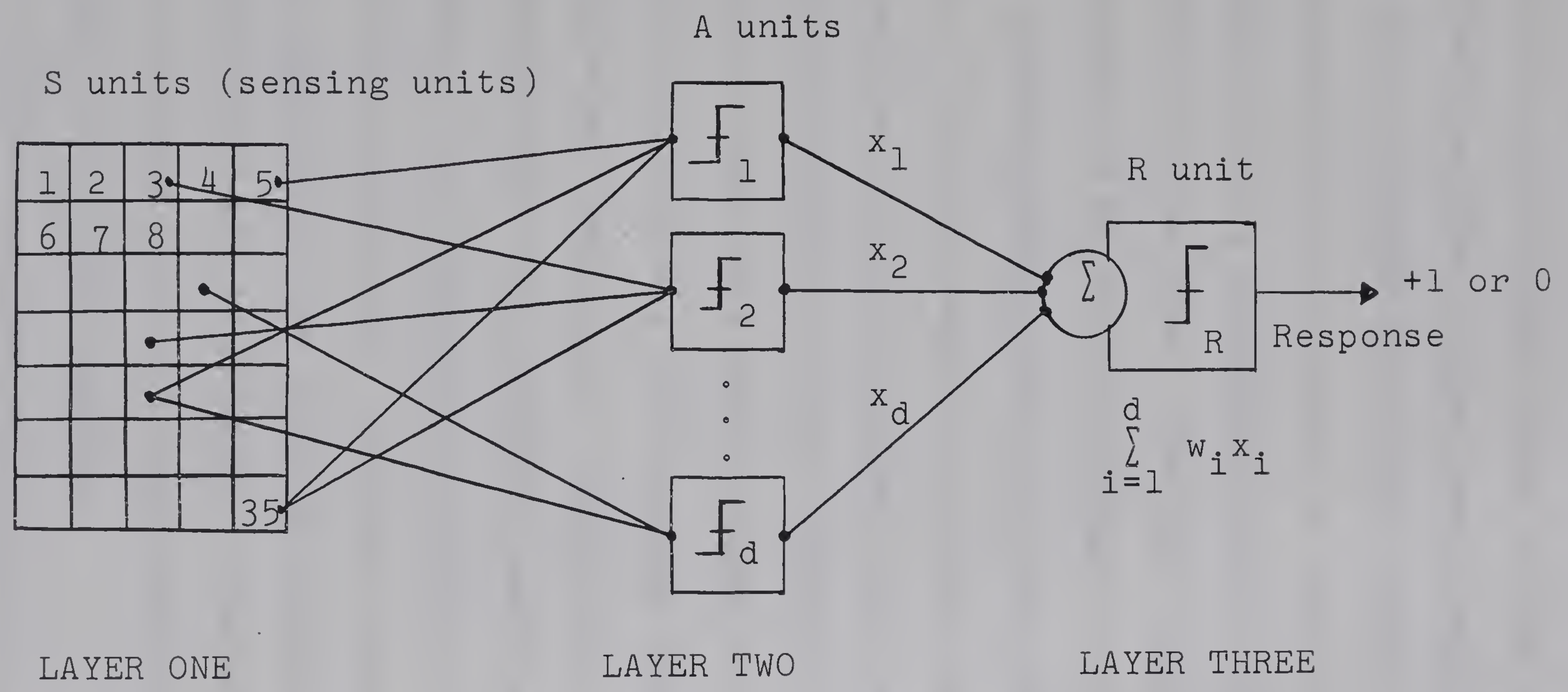


Figure 3. A schematic perceptron.

matrix). He found it necessary to normalize the center of density and average radius of the character-patterns to achieve success. Experiments performed on the TX-0 computer at M.I.T. showed that after twenty training trials a machine with two inputs, X and O and two R unit outputs, could recognize the two letters correctly 95 percent of the time. For the inputs O and Q the system obtained 80 percent performance after 100 training trials. By allowing modification of the random connections between the S and A units as well as the weights in learning, a machine with six outputs, could recognize, with 90 percent accuracy, any six character-patterns on the second trial if they were fairly close to the originals and after forty trials if they were sloppy. However the program was very slow, taking about seven seconds per character and its extension would be proportionately slower. In a subsequent program on a faster computer, the TX-2, with a 36 × 36 input matrix where the connections are <u>not</u> random, the complete alphabet was learned. The probability of recognizing a character was .94 in about 40 trials per character and characters were processed at a rate of four per second.

Another program which employs random n-tuples is that of Bledsoe and Browning (Uhr, 1966). This program is somewhat different from other "random" programs in that

it uses a storage and matching method rather than threshold and summing. The results of this program were an average of 35 percent recognition for hand written alphanumeric characters and an average of 78.4 percent recognition for hand block print, over five different choices of random n-tupling. It was found that pre-positioning of the patterns gave somewhat better results. The time for recognition was not evident.

All the programs taking the random approach have an underlying assumption that patterns can be characterized adequately by dividing them into arbitrary little pieces and then inspecting those pieces. This does not seem a promising approach when considering the concept of pattern, which is that it is a structure of individual parts whose meaning lies not in these parts but in the fact of their structure. In the random approach the programmer is designing a program to determine <u>how</u> to recognize patterns, which is an indirect approach in that the problem is to design a program to recognize patterns. Furthermore, at the present time, this method is impractical for general pattern recognition purposes. The time and space required for a neural-net model to be simulated in a computer does not lend it to productive efficiency; the trainable pattern classifiers, with randomly chosen inputs from a fixed grid, tend to be

intolerant of pattern variations and to require considerable time and care in their training. The random approach is valuable in that the results of the programs may suggest basic principles for pattern recognition and hence contribute to the design of productive pattern recognition programs.

## 2.3 The Template Approach

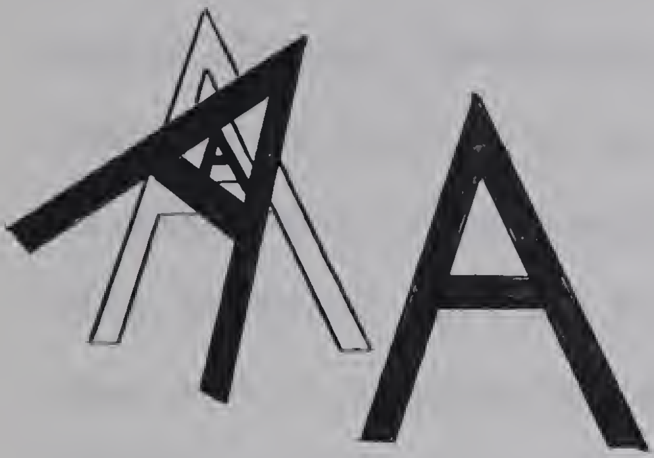
In the field of cognitive research, theories have been proposed that humans learn each pattern as a whole and that subsequent recognition is achieved through matching, (see Neisser, 1967). That is, a replica of a learned pattern is stored in the human memory as a canonical form or "template" and paradigms are recognized thru a matching search. Neisser, (1967), discusses the unliklihood of templates being used by humans and cites the results from a number of psychological studies in support of this view.

As a method for machine recognition of patterns the template approach has been fairly successful on controlled patterns, (Fischer, Pollock, Radack and Stevens, 1962). In this method the templates or prototypes of a set of patterns are stored in the machine. When a subsequent pattern is to be recognized it is compared with each of the templates and the name associated with the template that matches "best" is given to the pattern. One way to

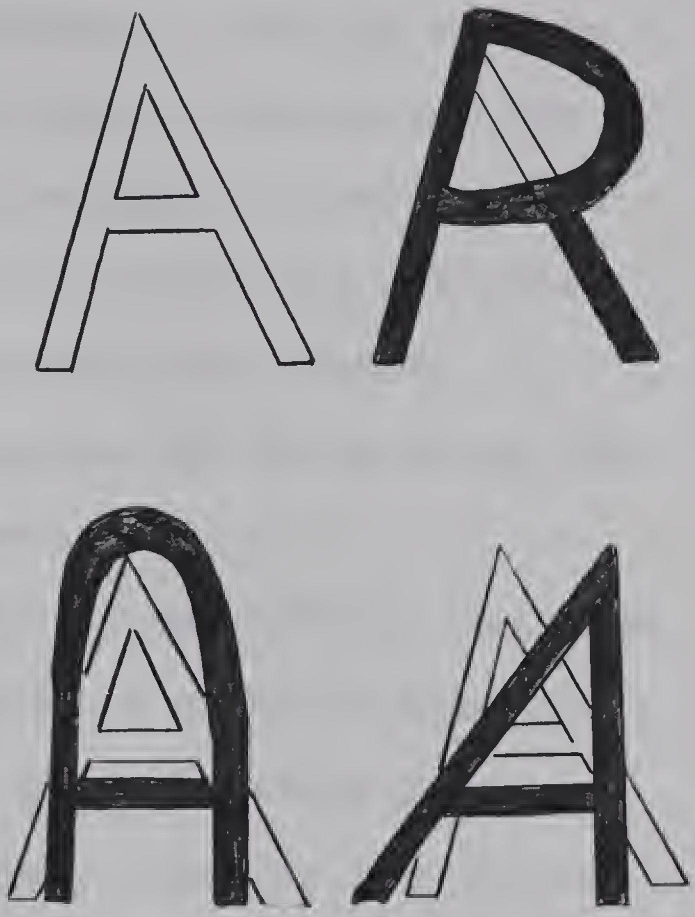
think of this is to imagine the pattern successively placed over templates composed of lights. The "best" fit corresponds to the minimum light within an upper bound, passing thru the pattern.

Clearly there are limitations to the template method. The patterns to be recognized must be of the same type font, size and orientation as the template and positioned exactly over it. Such noise as blurring, grain, low contrast, etc. may contribute to an incorrect recognition. Some of these limitations have been overcome in pattern recognition programs by inserting a level of information processing ahead of the template matching procedure. The pattern is shifted, rotated and magnified or reduced in order to put it into a standard form. Although this is an improvement over just matching such a procedure is still inadequate. Shapes may be compared successfully but letters are much more than just shapes. Even after adjustments a pattern may match a wrong template more closely than it matches a right one, (see figure 4).

The template approach is used in some special-purpose machines commercially. Often fairly rigid conditions on the patterns are necessary for these machines to operate effectively. For instance a commercial machine for reading numeric characters may require each character to be of a specific size, placed in a box, and composed



(a)



(b)

Figure 4. (a) Template matching cannot succeed when the unknown letter (color) has the wrong size, orientation, or position.
(b) Incorrect match may result even after adjustments. Here R matches A template more closely than do samples of the correct letter. (Selfridge and Neisser in Feigenbaum and Feldman, 1963).

in a specified manner etc. Such discrepancies as broken lines and patterns produced by poor quality ribbons or dirty type may cause trouble. Nevertheless the performance of these machines is impressive. The RCA machine (Hannon in Fischer et al., 1962) could distinguish the 16 alphanumeric patterns it was designed to recognize with an error rate of five errors per million patterns read and it recognizes 500 characters per second. Furthermore this machine is suitable for recognizing the complete English and Russian fonts.

Although no mention was made on the control over the characters read in the RCA machine in order to achieve such high accuracy performance, we would think that nearly perfect patterns were input. Heasley and Fischer (in Fischer et al., 1962) point out that even with controlled font there are many discrepancies in characters recorded on paper, including those caused by the recording machine, the paper quality, etc. He states that "each of the classes of machines have characteristic properties which determine character appearance". For example, in one class of high speed printers, there are four common types: stylus printers, matrix printers, drum printers, and chain printers. To pick just one characteristic from each, Heasley et al. states that stylus printers vary in overall character width; matrix

printers drop or add matrix elements when wires stick; drum printers have uneven vertical alignment which may result in loss of top or bottom strokes; and chain printers give uneven horizontal positioning which shaves left or right edges. Furthermore these defects are compounded by variations in hammer force, imperfect adjustment, ribbon wear etc.

As an approach to the problem of recognizing patterns, the template method appears somewhat limited for a really versatile character-recognizing facility since the font and quality of input must be well controlled. It appears that little can be done to improve template matching, other than increasing the resolution of the optical reader, rereading lines when poor quality is encountered, and adding to the number of fonts in the machine's memory. Little or no insight into the general pattern recognition problem can be gained thru these attempts for improvement. From these observations on the template approach, it appears that a pattern, which may take many physical forms, cannot be treated literally as a whole for the purposes of recognition. This indicates that some form of parsing and examination of the parts is necessary for effective recognition.

## 2.4 The Features Approach

In this approach it is assumed that a pattern is a

collection of basic parts which may be used to identify it. Recognition is achieved by noting the set of basic parts of an unknown pattern and comparing that set with the typifying sets of learned patterns.

The features approach is supported by research results in neurophysiology. Several studies made on the visual cortex of animals (see Barlow, Burns, Heron and Pritchard, Hubel and Wiesel, and Young in Uhr, 1965) give evidence that animals have brains which are organized to receive specific configurations of optical stimuli. The evidence suggests a built-in system for "seeing" the visual stimuli in a manner which contributes to the functioning of the animal.

Several programs (Narasimhan, 1966; Eden, 1962; Grimsdale, Sumner, Tunis and Kilburn in Uhr, 1966; and Uhr and Vossler, and Selfridge and Neisser in Feigenbaum and Feldman, 1963) have been built, for either pattern recognition or pattern reproduction, which exhibit some of the aspects of this approach. As a rule these programs show a high degree of flexibility in that they can deal with patterns of varying size, font, orientation and position.

The program by Grimsdale, Sumner, Tunis and Kilburn, is an apt example of the features approach. In this program there are a number of stages. First the pattern

is transferred from paper to the store of the computer. The pattern is then examined by a programmed scan to produce descriptions of segments of the pattern (see figure 5). The scan also allows for noise such as figure gaps, dirt on the paper, etc. Next, in the assembly stage, the segments of the pattern obtained in the scan are analysed and connected into larger pattern parts; a description is given of the length and slope of straight-line parts and the length and curvature of curved parts. The scan and assembly sections of the program produce the "statement", which is a complete description of the pattern. This description is independent of the orientation and size of the pattern since the lengths of the various parts are given relative to one another. The statement is, in effect, a coded representation of the pattern which is essentially a one-dimensional representation (or pattern) of a two-dimensional figure.

When the program is called upon to recognize an unknown pattern, a comparison is made between the statement describing the unknown pattern and the statements already stored within the computer. An automatic classification system is used so that not every stored statement need be examined. This arranges the stored statements into classes according to common features. As an example, all statements describing figures with four ends

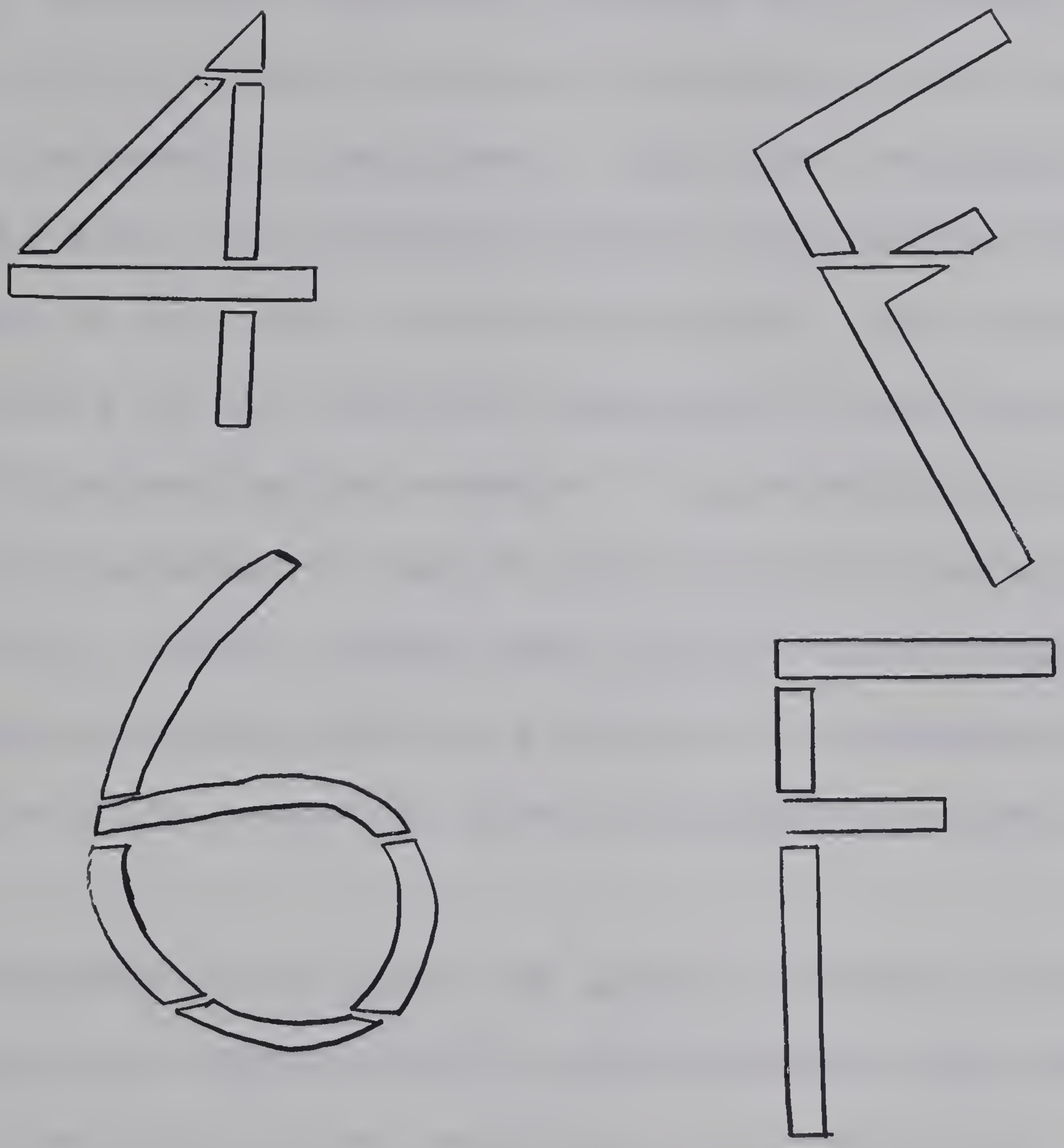


Figure 5. Segmented patterns. (Grimsdale et al. in Uhr, 1966).

would be grouped together so that a statement describing an unknown pattern with four ends would be compared initially with the statements of members in the "four ends" class.

A measure of agreement between the "unknown" statement and the "known" statement is used as the criterion for a successful comparison. This must be significantly greater than the agreement between the unknown statement and any of the other stored statements. The relative importance of the features expressed in the statement and used in assessing the measure of agreement, may be stated by the programmer or may be determined by the system thru learning. In the latter case various classification systems are tried and the systems are reordered depending upon the time needed to find the required stored statement.

Grimsdale's program was given a library of standard patterns for capital letters and numerals and could recognize all of them regardless of orientation. Using the same library of standard patterns the machine was capable of recognizing a range of other pattern of varying size, form and thickness. However thickness sometimes caused problems which could be solved by increasing the resolution of the scanner or reducing the thick patterns to thin patterns before scanning. Small breaks in the pattern, ragged edges and dirt caused no

problem. If a pattern could not be recognized a list of possible names together with their degree of certainty was printed.

The program fails with patterns which contain a large amount of detail. This is attributed to insufficient resolving power in the scan. Due to the nature of the scanning program the whole pattern is inspected but only information concerning the edges is recorded. The scanning procedure takes a large amount of time and it is felt by the designers that the scan examines the pattern in too great detail. This may be alleviated by having the scanner indicate general trends and thereby eliminate some unnecessary procedures. The program is implemented on a medium speed digital computer. It has approximately 4000 instructions, and the time to recognize a pattern is of the order of 60 seconds, depending on the complexity of the pattern and the size of the library.

Another program utilizing the features approach and presenting some interesting ideas is Narasimhan's work, 1966. His program uses generative grammars for bubble chamber pictures and hand printed letters to allow the computer to construct particle tracks and letters. He felt that an adequate means of describing patterns is necessary not only for their generation but

also for their recognition. We would agree with Narasimhan; but with the reservation that perhaps separate types of description are needed for the two tasks of generating and recognizing. Although one grammar may suffice, it may not be so efficient for recognizing as for generating, or visa versa. For practical purposes perhaps a more specialized grammar should be tailored to each particular task.

In Narasimhan's generative grammar for hand printed letters, he has chosen particular shapes of various sizes, which have numbers associated with certain points, as his primitives. The numbered points are vertices or joining points. In his grammar he employs a symbol to denote a particular shape. To indicate the conjunction of two shapes he catenates their symbols and provides a list of joining points. In this way successively larger units may be generated. For example suppose v is a straight line with 1, 2, 3 for top, middle and bottom respectively and d is semi-circle oriented vertically and concave on the left with 1, 2 for top and bottom, then the pattern D may be expressed $v \circ d(11,32)$ (see figure 6). Usually the list for a pattern contains alternatives, that is Narasimhan gives

$D = v \circ d(11,32;) | r \circ d(11,32;)$ .

Narasimhan's program was successful in generating

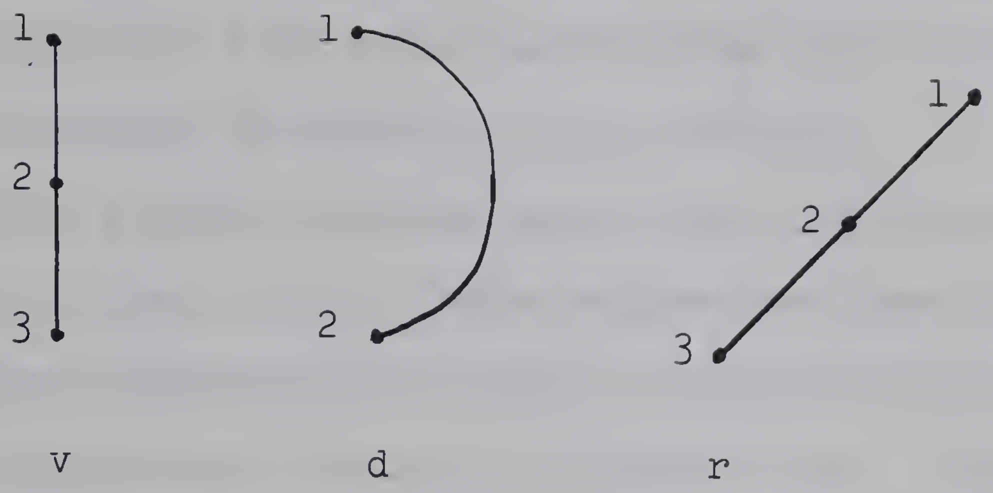


Figure 6. Examples of Narasimhan's primitives. (Narasimhan, 1966).

letters from the statements in terms of the grammar described above. However since no attempt at using the grammar for recognition purposes was made, this aspect of its use is inconclusive. The importance of this paper is its explicitly stated grammar which may be used to form one-dimensional statements of two-dimensional patterns, thus reducing the two-dimensional information into the more tractable one-dimensional form while preserving the distinctly two-dimensional relations in the pattern.

Still another program supporting the features approach is that of Eden, 1962. This program was developed to reproduce handwriting in order to explain that human act by an appropriate theory or by simulation. Identification by a generative procedure leads to a clear definition of the set of permissible patterns. The class of acceptable patterns is that set which can be generated by the rules operating on the primitive symbols of the theory.

Eden employs a generative grammar composed of defined primitives to "write" words. That is his program performs an analysis of a written word and then reproduces the written word. This program was successful in its forgery of handwriting. An implication of his generative program for handwriting recognition is that some provision must be made for resolving ambiguities, such as the use of word dictionaries.

Some special purpose machines employing the features approach have been developed commercially. Farrington Electronics Inc.'s Selected Data Page Scanner (in Fischer et al., 1962) is one which uses a stroke-recognition technique for character-pattern analysis. This technique is based on the way characters are formed by line segments. It is able to reread characters of marginal quality until recognition is achieved or until a reread limit is reached. This machine recognizes capital alphabetic letters, arabic numerals, common punctuation marks and special control symbols which can be output onto punch cards, magnetic tape or paper tape at the rate of up to 240 characters per second.

Finally we shall mention briefly the program by Uhr and Vossler, in Feigenbaum and Feldman, 1963. This program combines all three methods. The program's random aspect is its ability to develop operators which are accepted or rejected according to subsequent utility. These operators are essentially partial templates and their application is a matching process with portions of a pattern (see figure 7). The operators may be thought of as the features to be considered. In this program unknown patterns are presented to the computer in discrete form, as a 20 × 20 matrix of zeros and ones. The program generates and composes operators by one of several random

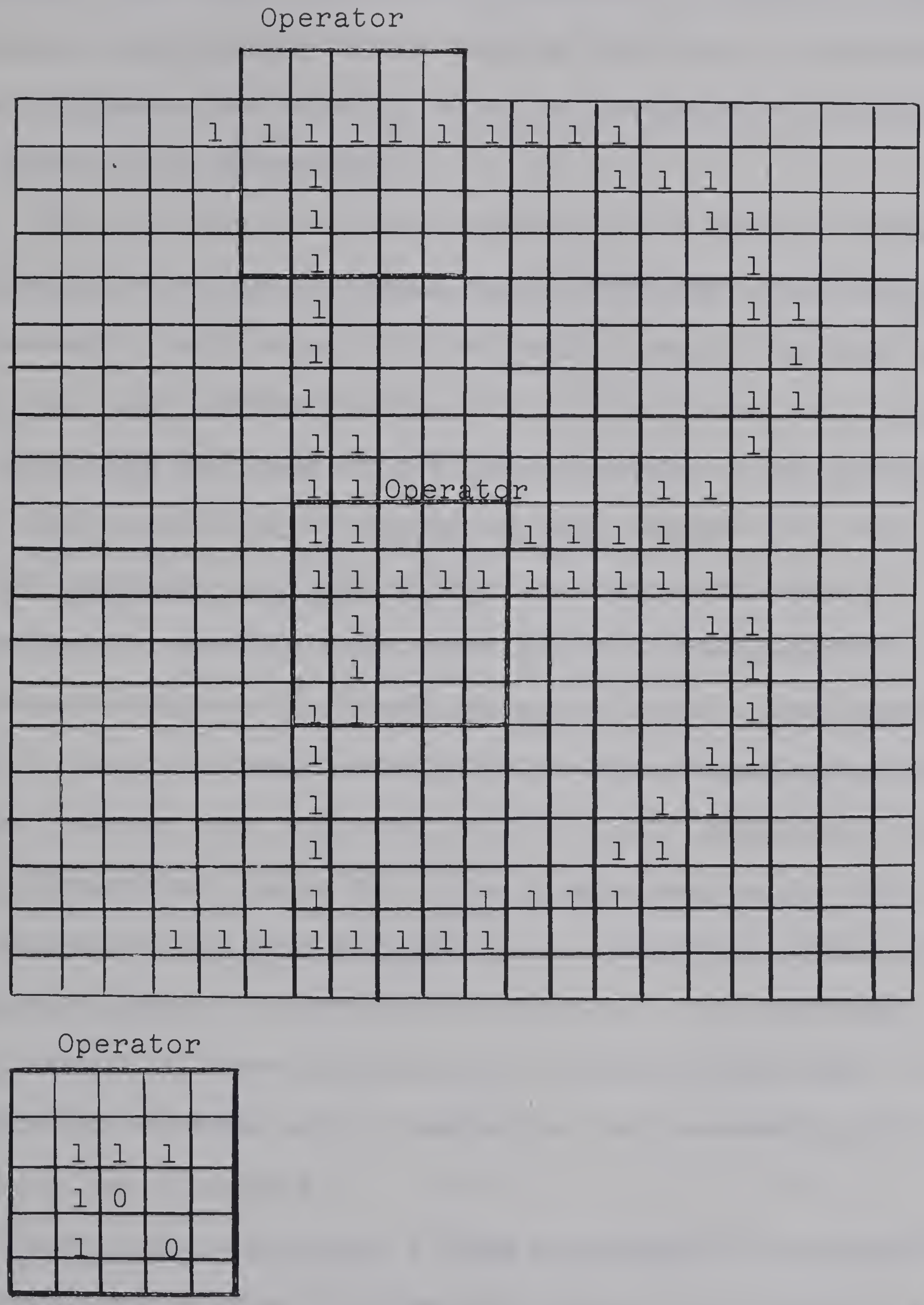


Figure 7. An operator and an example of its use. (Uhr and Vossler, in Feigenbaum and Feldman, 1963).

methods, and uses this set of operators to transform the unknown input matrix into a list of features. Alternately, the programmer can specify a set of pregenerated operators in which he is interested.

The features are then compared with lists of features in memory, one list for each type of pattern previously processed. As a result of similarity tests, the name of the list most similar to the list of features just computed is chosen as the name of the input pattern. The features are then examined by the program and, weights for each of the features are then raised or lowered in value, depending on whether they individually contributed to success or failure in identifying the input. The adjustment of weights leads eventually to discarding operators which produce poor features, and to their replacement by newly generated operators. The program may begin with no operators and generate them at a fixed rate until some maximum number of operators is reached. The continual replacement of poor operators by new ones then tends to produce an optimum set of operators for processing the given class of inputs.

The program achieves a high percentage of recognitions over various classes of patterns; however it requires a large amount of time, i.e., 40 seconds when each pattern is compared with ten possible patterns in memory. The

time was greatly reduced, (to 1 second), when the weights were not changed. But then the program was weaker since it did not determine the best operators. Also the program required several passes thru a class for the purpose of determining the operators. It seems that the operators would be somewhat limiting in the variety of line-thickness they could handle due to their template nature. In a sense this dependence of the operators on the learned set is the programs power as well as its limitation. The importance of this program lies in its ability to develop operators which arise from the patterns and thus are sensitive to the patterns. These (weighted) operators may be thought of as determining those features which are important to the class of patterns under consideration for the purposes of their identification.

The power of a features approach program lies in its capability of being <u>extended</u> to process new classes of patterns. However each extension may be an inadvertent complication, instead of a simplication, and would represent an additional burden of time and energy on both programmer and computer. Some kind of early decision concerning the pattern as a whole and directing attention to those features which are pertinent, may allow for extensions which will not appreciably increase the time.

## 2.5 The Interrelationships of the Approaches

It will be noted that the random approach is essentially the template approach but where many templates are "learned" and the result, ideally, is a function or set of functions depicting the correlations of all the templates for one pattern and the variances among the templates for diverse patterns. While the template approach can recognize only one configuration of each pattern, the ultimate goal of the random approach is to recognize all configurations of a pattern after having experienced a sample of configurations.

The features approach is a hybrid of the template and random approaches. It employs templates in the sense that it is preprogrammed to detect specific arrays in input patterns and it is random in that for a particular pattern it is not known what sub-set of specific arrays will be applicable in the majority of configurations of that pattern. Learning is achieved in the sense that a sub-set of specific arrays or features is determined for each standard pattern, where it is assumed that the features will be dominant or highly likely to appear in subsequent configurations of the pattern. Obviously the choice of features is critical. In contrast to the random approach in which the program essentially chooses the features, in the features approach they are chosen by the programmer using his knowledge of the class he

is attempting to recognize.

## 2.6 The Approach Used

When a botanist is classifying a flower he does not chop it into arbitrary parts and shake them through a self adjusting sieve. Neither does he attempt to match it as a whole with a known flower. Rather, he inspects those parts of the flower which he has come to know are pertinent to the classification of flowers.

If one is attempting to build a machine which will classify a set of patterns it is reasonable to use all that can be observed about the set to determine how the patterns might be classified. Furthermore the ideal program is one which will recognize patterns regardless of their line-thickness, size, form, orientation, position and the presence of noise (within reasonable limits). The features approach appears to be the most successful in recognizing patterns independently of these variations, i.e., "uncontrolled" patterns. For these reasons it is felt that the features approach is the most practical for implementation on a general purpose computer and will also lead to fruitful results. Since we hope to achieve a program which will accurately recognize visual patterns presented in their variety of aspects with minimum programming and computer effort, the features approach was chosen for building a pattern recognition program. From

hereon the word program (or machine) shall imply the features approach. This approach is supported by Neisser, 1967; Uhr, 1965; Newman, 1961; and Greanias, in Fischer et al., 1962.

As suggested by Newman, 1961, and Narasimhan, 1966, we have limited the class to be recognized. In essence the program "knows" the class it is recognizing and has a set, in the psychological sense, for that class. It is assumed a higher program in a hierarchy of programs or the computer operator has recognized the class and directed the efforts for further categorization to this program. The class is printed alphanumeric characters and to investigate this class we have used a sub-class. By working with a finite set of patterns we can hope to find general principles concerning the correlations among configurations of a single pattern and the variances among the set of patterns, thru inspection, which will lead to general principles concerning the class in which the set is imbedded. (These general principles are depicted by our choice of features and tested thru the program use.) Then one property of the features approach, that of extension, may be utilized to allow the program to embrace the whole class. By developing a program for a finite set and applying it to members of an extended set, we can gain insight as to the limitations of the program

which may point to the manner in which the program should be extended. The sub-class for our program development is capital alphabetic letters.

# CHAPTER III

# PRELIMINARY INVESTIGATION FOR A PATTERN RECOGNITION PROGRAM

## 3.1 Introduction

The stages of a pattern recognition system may be defined by its functional divisions. The most natural divisions are: input, normalize, describe, and classify. The two latter stages are grouped together into what is called the "define" stage, where the define stage may be in one of two modes, the learn mode or the recognize mode. The input stage transfers the pattern from the external environment to the machine's internal environment. The normalize stage puts the pattern in an optimal form for recognition. This may involve filtering noise, filling in gaps, shifting or positioning the pattern, adjusting or determining orientation, changing its size, etc. The define stage processes a pattern and determines its coded representation; this is essentially a translation of the pattern, from two-space to one-space. In the learn mode of the define stage the coded representation is stored in memory for future referral. In the recognize mode the coded representation is "compared" with the coded representation in memory and identified.

In this chapter we discuss the stages of a pattern

recognition system with an emphasis on the results of the input stage and the implications of these results for a pattern recognition program, i.e., the define stage of the system. We then present a proposal for an input device structured after the human fovea and describe the linear representation of two-dimensional patterns which we used. Our aim is to develop an accurate program for uncontrolled capital alphabetic characters which is fairly fast.

## 3.2 Input

### 3.2.1 The General Problem

The general problem of the input is to transfer an external representation of a pattern into a computer code while maintaining the essence of the pattern. This is generally accomplished by using an auxiliary light-sensing device, which registers for different intensities of light. Some such input devices are a field of photocells, an optical scanner and a flying spot scanner.

The use of each of these types of input device can result in the same internal representation of a pattern. We will assume that this initial internal representation is a binary matrix which is called a pattern matrix, input matrix or matrix in subsequent sections. The input device, that is most often referred to, is the photocell array as it is amenable to hand simulation. Such an

array is called a grid, or sensing grid, while the photocells are referred to as elements, activation elements, sensing elements, or grid elements. Any element which registers dark (the presence of a pattern portion or noise) is called an activated element and causes a one to be placed (registered) in its corresponding matrix location. All other matrix locations are zeros.

Often when a method of input is presented in the description of a pattern recognition program little or no justification or comment is made why a particular input method is chosen. Some comments are: "We start with a 10 × 15 photocell mosaic (this size being chosen because of immediate availability)," (Bledsoe and Browning in Uhr, 1966, page 302). "Input of free-hand characters was obtained by drawing them on the computer-controlled display scope with a light-sensing pen. A character could thus be represented on a 64 × 64 bit matrix." (Roberts in Uhr, 1966, page 295.)

Some comments on input with partial justification are: "Unknown patterns are presented to the computer in discrete form, as a 20 × 20 matrix of zeros and ones". - and later in the paper - "The size of the overall input matrix has also been chosen with the requirements of pattern perception in mind. Good psychophysical data show clearly that when patterns of the complexity of alphanumeric letters are

presented to the human eye, recognition is just as sure and quick no matter how small the retinal cone mosaic, until the pattern subtends a mosaic of about the 20 × 20 size, at which time recognition begins to fall off, in both speed and accuracy, until a 10 × 10 mosaic is reached, at which point the pattern cannot be resolved at all". (Uhr and Vossler in Feigenbaum and Feldman, 1963, pages 252 and 266.) Arkadev and Braverman, (1967, page 12), say: " - we have used fields comprising comparatively few (60) elements, so that numerals and letters depicted in such fields have a "grainy" structure and differ from the ones that we normally see. ...the particular way in which the field is divided and the shape of its elements are rather immaterial, since we can easily conceive a field of triangular, diamond-shaped, or hexagonal elements. We use a square net, the simplest one."

Uhr and Vossler's statement contains a reference to psychophysical data but he does not give the source of his reference. Furthermore no mention is made as to the area of the grid elements, ("grain size"), or the thickness of the line, ("line-size"), which he is using. If the sensing grid is not divided into a small enough grain-size with respect to line-size the computer will not obtain enough information to determine any consistent distinctions between different patterns. On the other hand if the

grain-size is very small with respect to line-size more information will be input and manipulated than is necessary for, say, character-patterns.

Arkadev and Braverman's statement concerning the shape of the elements in the sensing grid is misleading. The size and the shape of these elements is important; they determine what the computer "sees" as the pattern, (see figure 8 and Appendix A). The shape of the sensing elements and the coded representation of them in the computer may be biased for some attributes. That is some features may be easily represented in a consistent manner while other features will have a variety of coded representations. The "consistent" features will have a higher probability of being recognized correctly. For instance, a sensing grid of square elements arranged in rows and columns is biased for horizontal and vertical straight lines which will more likely be recognized correctly than say curves or diagonal lines. The reasons square element sensing grids are usually used are the availability of such devices and the ease in coding them, (some shapes of elements in a grid are difficult to code). Computers naturally lend themselves to codes for two-dimensional arrays which result in a matrix and other representations of a two-dimensional pattern in a computer may be difficult to achieve and cumbersome to manipulate.

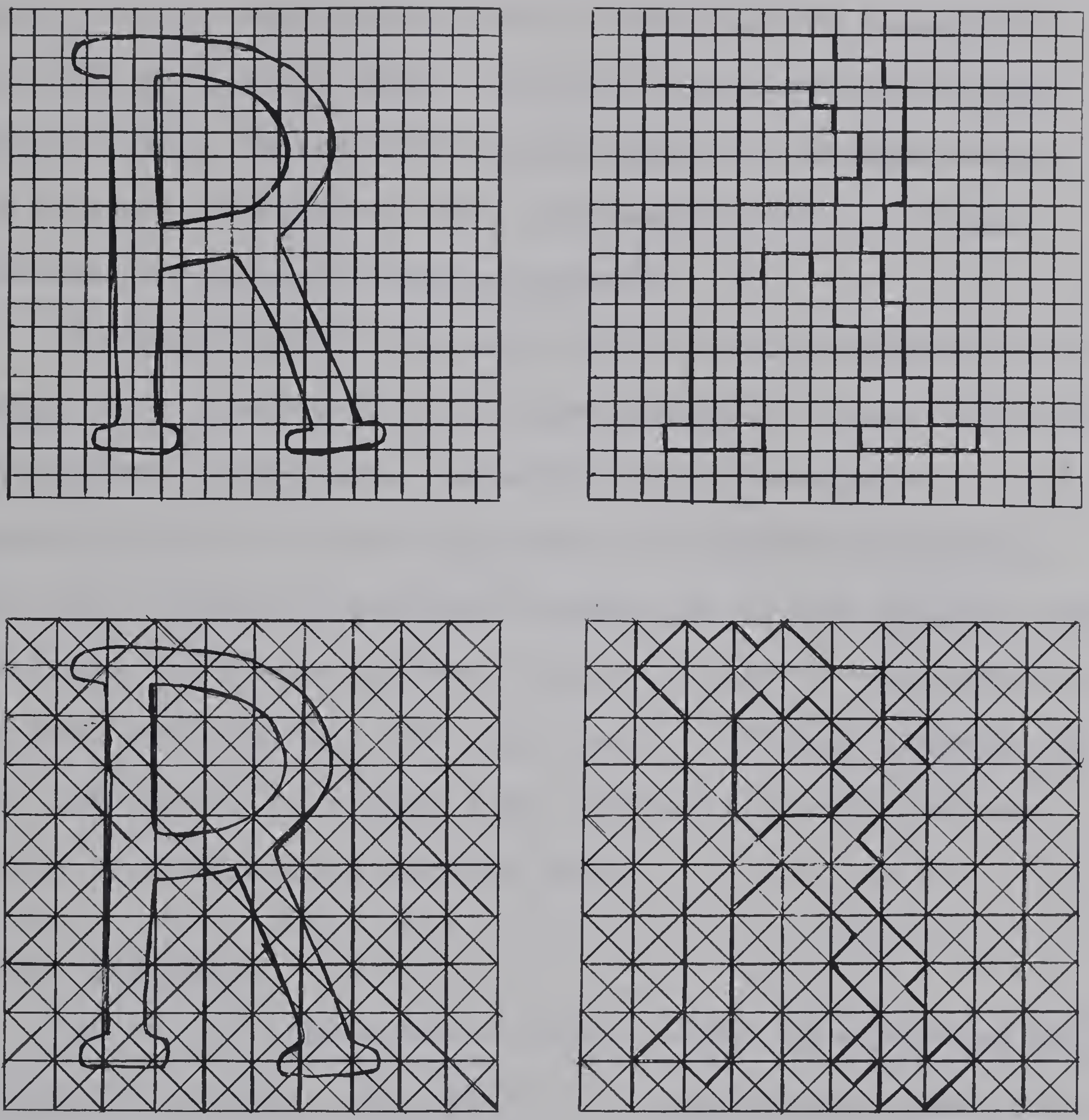


Figure 8. Two sensing grids, one with square elements and one with triangular elements, and the registered results when each is presented with the same character pattern.

Other input considerations are pattern quality, paper reflectivity, ink density, etc. These may be thought of as causing forms of noise and will not be discussed here specifically. Merry and Norrie, 1961, and Heasley et al. and Greanis in Fischer, 1962, present details on these problems in character representation.

The input problem on which we will concentrate is the design of a sensing grid, in particular the shape, size and arrangement of the grid elements. To investigate possibilities for a sensing grid it is natural to study the most successful pattern recognition system that we know, the human eye-brain system. In this study it will not be our goal to duplicate the input device of this system, the eye, but rather to obtain some information which may be useful in designing a sensing grid.

## 3.2.2 The Eye

The eye is organized to direct light to a sensing grid called the retina. The retina is connected with the optic nerve by way of a record layer of nerve cells. The optic nerve carries the "coded" pattern to the brain where learning and recognition are achieved.

The retina (sensing grid) is described here after Polyak, 1957. The retina is a "mosaic" or "pavement" of nerve cells and may be considered in terms of regions demarked by concentric circles. The outer region of the

retina is generally more sensitive to diffuse light and is composed of two kinds of photoreceptor cells, intermixed, the rods and cones, where the rods predominate in number. The rods are used primarily in scotopic (dark-adapted) vision and the cones in photopic (light-adapted) vision. The next region is called the central area; here there are more cones than rods. In the center of the central area is a circular, flat-bottomed, bowl shaped depression called the fovea. The fovea is composed almost entirely of cones except for a few rods interspersed on the upper parts of the depression. The central fovea, the "flat-bottom" of the depression, is called the foveola; this area corresponds to direct vision in the axis of visual acuity and contains only cones.

The number of cones in the central fovea is absolutely and relatively greater per unit area than anywhere in the extrafoveal regions. They are closely packed into a solid formation, without any gaps, (except for the extremely narrow spaces between the adjoining cones); the entire formation here resembles an unevenly distributed mosaic. The elements of this mosaic, the individual cones, are mostly hexagonal, less frequently pentagonal in shape. They are arranged in fairly distinct rows, somewhat arching and cutting across one another in three definite ways at approximately 60°. The pattern is not mathematically

regular, but shows numerous irregularities and deviations from a theoretically postulated arrangement. "The arrangement of the "grain" of the central foveal cones as described indicates small preference, if any, for certain directions, lines, or shapes". (Polyak, 1957, page 269.)

### 3.2.3 Input Parameters and Bounds

There are essentially two sets of parameters that determine what information about a pattern the machine obtains. These sets are (1) those parameters which define the sensing grid and (2) the parameters which delimit the pattern. The values of some parameters in one set are dependent on the values of parameters in the other set. The sensing grid parameters are - the number of grid elements, their shape, size, threshold and separation. The pattern parameters are - pattern size, line size (thickness of line), space size (dimensions of space bounded in whole or in part by lines), and skew allowance (how much a pattern may be rotated from an optimal position).

We have assumed that patterns will be magnified and read in constant, optimal illumination and that all sensing elements have the same threshold. The sensing elements may be thought of as photocells each of which sends a one signal when light is not sensed and a zero signal otherwise.

We chose the hexagonal shape for our sensing grid elements mainly because a hexagonal element, photo-sensing

mosaic will have small bias, if any, for certain directions, lines, or shapes. The hexagonal is the end-shape of photoreceptor elements used for acute vision in animal eyes and is the result of aeons of selection from what was available in nature. Not only is the human fovea a hexagonal mosaic but this same shape also occurs in other animal eyes, both vertebrates and invertebrates, which are highly developed.

The remaining sensing grid parameters - size, separation and number - are interdependent with the pattern parameters - pattern-size, line-size, space-size and skew allowance. A pattern presented to a grid must be of adequate size in relation to the grid for its distinguishing features to be registered or "seen". A single line must be wide enough to equal or surpass the covering-threshold of at least one sensing element. For two lines to be resolved as such the space between them must be registered as a space. A straight line must be long enough to activate a minimum number of elements in a sequence, etc. In general characters on documents are rather small in relation to the physical size of a grid element it is possible to make. This is particularly true for hand-simulated grids. Assuming we have a specified grid size and we can measure the approximate size of characters in a document, we can determine the magnification necessary to obtain resultant

characters which are adequate for resolution. That is we can determine the minimum requirements on a pattern for just adequate resolution. The minimum magnification was found by considering the threshold, the size of the grid element and the space separating the grid elements, with respect to the minimums of line-size, space-size and line length, which we wish to detect, of the original pattern.

### 3.2.4 The Mechanical Fovea and Coding

The following description is of a proposal for the configuration and operation of an electronic sensing grid. Actually the grid and its operation are simulated by hand, see Appendix A.

The input grid that we use for our program is somewhat like the foveal mosaic and we call it a "mechanical fovea". It is composed of closely packed hexagonal shaped "sensing" elements, (h-elements), each of which "sends" a one or zero to a matrix location in a matrix in the computer, depending on whether its covering-threshold is surpassed or not. The diameters of each h-element are nearly identical and measure about 4.5 mm., the separations between adjacent h-elements are about .1 mm., and the covering-threshold is about 10% of h-element area. The mechanical fovea has 21 h-elements in a row and 25 rows, (see Appendix A).

Since the mechanical fovea does not lend itself directly to a one-to-one correspondence of h-elements with

locations in a matrix, a coding scheme was developed. We chose to represent the mechanical fovea internally as a matrix because matrices are easily manipulated. Although this adds some bias for vertical and horizontal lines, it does not seriously affect the model. We did attempt to preserve oneparticular property of h-elements in our coding. This property is the fact that each h-element has an identical relationship with all of its adjacent h-elements; each h-element has a common face with exactly 6 adjacent h-elements. (In the usual square-element grid each element has a common face with 4 adjacent elements and a common point with 4 other adjacent (?) elements.)

The coding scheme can be described as follows: odd numbered grid rows can register ones in only odd numbered columns in the matrix and even numbered grid rows can register ones in only even numbered columns in the matrix where the matrix is orginally all zeros. All zeros in a matrix row with a one on both sides of it in the row are changed to one and similarly for columns. This "in-between" one represents the common face shared by two h-elements. It is evident from the coding and the hexagonal configuration that the matrix to represent a grid will be larger than the grid, i.e., our input matrix has 25 rows and 40 columns. The resultant matrix of a pattern gives a horizontally widened internal representation

and considerable redundant information. The magnification used for most cases is about twenty five times the original pattern size. See Appendix A for a coding example.

### 3.2.5 Implications of the Information from the Input Device for Designing a Pattern Recognition Program

The exact size line thickness, relative orientation of lines, contour of curves, and position of the pattern on the h-element grid will affect the resulting coded matrix. Since we wish to ignore these variations, we should look for features that are independent of them, but which characterize the patterns in the set. The relative size and location of closed holes and the orientation and depth of open holes would seem to fit this requirement. (See Section 3.4.3 for a more complete description of these "holes".) That is, the delimited spaces in a pattern can be used to depict the pattern and the presence of these spaces is not affected by the variations mentioned above.

In addition, since there is information in the matrix that does not contribute to our objective, it would be efficient to make use of general descriptive measures of the pattern first. Then, if the possibility for a particular feature became evident, further more precise measures could be taken to verify the presence of that feature.

Altogether, the implications of the information from the input device suggest a pattern recognition program which assumes some variation in the coding of different examples of what we wish to consider as the same pattern. Furthermore, a hierarchical approach is indicated by the need for efficiency.

## 3.3 Normalize

The purpose of the normalize stage is to put the pattern in an optimal form for recognition. In our program we do not attempt to filter noise, shift the pattern, change its size etc. We fill in gaps to the extent of completing the hexagonal grid to matrix coding procedure. That is wherever a 101 occurs in a row or column we change it to 111. The rows are done first, then the columns. Furthermore we determine the pattern height and width in terms of maximum number of matrix locations between and including the farthestmost ones in a column (for height), and in a row (for width). That is if 10001 is the longest column string bounded by ones it would give a value 5 for pattern height.

As far as orientation is concerned only properly oriented characters, or those only slightly skewed, can be

recognized correctly. No pre-orientation is done by the program. It is reasonable to assume proper orientation since humans normally read this way and otherwise encounter added difficulty. The normalize procedure here is effectively the completion of matrix coding and the determination of pattern size.

## 3.4 Define

### 3.4.1 The General Problem

The purpose of the define stage is to process a pattern matrix to determine its coded representation and to either store the representation or use it to classify a pattern matrix. The problem is to determine an "ideal" representaion for a two dimensional pattern. The ideal representation need not be an exact depiction of any one of the forms of the pattern but should be easily compared with the ideal representations of all the patterns in a class and be different enough to conclusively match with only its canonical ideal representation. In the features approach, which is the approach we are using, the ideal representation is generally a one dimensional pattern with components typifying the features. That is, the problem becomes the mapping of a two dimensional pattern into one dimension while retaining the distinguishing features of the pattern.

It is clear that the representation depends on those

features chosen to define, or describe, a pattern. We will present a one dimension representation for optical patterns which is based on the features we have chosen to define character-patterns. Our representation is constructed primarily for purposes of recognizing patterns rather than recording them for future reproduction. The representation could be used for reproduction but it may not be so efficient as a representation designed for that purpose.

The following sections will discuss the criteria for selecting the features in our representation, describe the features and their limitations, present the representation, and describe how it should be used in learning and recognizing patterns.

## 3.4.2 Criteria for Selecting Features

A sensing grid which yields a binary matrix of size $n$ rows by $m$ columns provides $2^{m \cdot n}$ possible matrices, assuming independence and equal probability of the binary entries in the matrices. We will say such a matrix contains $m \cdot n$ bits of information and has $2^{m \cdot n}$ possible states, see Osgood and Wilson, 1961. By representing a pattern as a series of features we can reduce the amount of information, $m \cdot n$ bits, to an amount corresponding to the number of possible features. That is, if $k$ is the number of possible features, a pattern is represented by $k$ bits of information and the number of states of the pattern

representation, assuming independent, equally-likely features, is $2^k$ . Hence given k bits of information we can represent $2^k$ distinct patterns with $2^k$ binary vectors of k length. In practice it is desirable to allow more than one representation for each pattern. If we let the number of possible vectors per pattern be four, we will have $2^{k-2}$ states representing distinct patterns. From this it appears that the more features we choose the more characters we can represent. The number of features, k , must be large enough so that the number of characters we wish to recognize, $C_N$ , is less than or equal to $2^{k-2}$ , i.e., k is selected such that $C_N \leq 2^{k-2}$ , or $k \geq 2 + \log_2 C_N$ .

On the other hand if we choose too many features we will have many more possible vectors than are needed and probably an inefficient program; unless (1) high level branches eliminate detailed processing of all features and (2) the matching process for recognizing a pattern is not affected by the number of possible representations. If these conditions are met considerably more features than are theoretically necessary can be included in the representation without adding appreciably to the processing time.

The features we choose will be pattern-set dependent. We must choose those features which will enable distinctions

among different characters yet not among a single character's various fonts (we use the word font to include hand printing as well as styles and sizes of mechanical printing).

To limit the number of features, we choose those which appear in many characters but never in the same combinations. Some features which are not necessary for definition of most characters must be included as they will be the only distinguishing feature between two characters; (we call these critical features). We would never include an attribute which is common to all characters, but rather we would use such an attribute to aid in determining the presence of a feature, if possible.

The features we choose should be invariant to noise, to detailed differences of fonts, and to peculiarities resulting from grid placement. The features should be simple in nature so that the processing of a pattern matrix to determine those features present will not be unduly involved and time consuming. They should lend themselves to tests at a high level which reveal whether they are conclusively absent or possibly present.

Obviously, limiting our pattern set makes it easier to determine the distinctive features needed in the program. Nevertheless we attempt to choose features which are of a general nature in that they may be effective in distinguishing patterns in an extended set.

In summary, the features are chosen so that there is some redundant information, and the features are simple and general in nature, tend to involve the whole pattern or large pieces of it, are reasonable to program and allow efficient processing.

## 3.4.3 The Features Selected

We can classify character features into two mutually dependent sets, linear and spatial. Linear features are lines or contours while spatial features are the spaces defined by contours and are what we have termed "holes" in a pattern. A character may be thought of as an ordered set of lines in two-space, subdividing the space and delineating sub-spaces.

We select only one linear feature, i.e., straight lines. This is expanded to a class of six features by considering orientation and location in a pattern, where two orientations with three locations each are used. The straight lines are to be nearly as long as the pattern width or height; they are vertical or horizontal, and are located in the left, right, center and top, bottom, middle of a pattern, respectively. These straight lines were chosen because they appear in a consistent matrix configuration, they are easy to determine, and they are large pieces of many characters.

We select one spatial feature, called holes, which is

propagated into a class of many features by including some attributes of holes. The attributes are "openness" and "closedness", orientation and size of opening, depth of open hole, and location and size of closed hole. Combinations of these attributes result in twenty-five features which are described later in this section. First we will explain the advantage of observing the spaces within contours. One might say that you have to find the contour to define a hole so why not use the contour. This is not true. In fact we do not even care what shape the contour has and we never determine it. We can assume certain facts, including the shape, because our pattern set is limited to a certain class - the capital alphabetic characters. These patterns have one common property, namely continuity; i.e., they can each be traced over with a pencil without lifting the pencil, even though some lines may be traversed more than once. We use this property implicitly, to define holes and determine their attributes from a small number of points. Thus in our program we employ the characteristics of holes rather than their shape.

A few critical features were selected. These features do not belong in either of the above classes and have a limited use. One critical feature is termed "density" and it compares the number of active entries (ones) in a matrix location with the number in the complement. This yields

a class of six features. Another is called "points" and detects certain small details inserted in a hole or extended beyond the major portion of a character; this provides a class of two features. The last feature is an insurance feature to prohibit multiple identification for a single line and to indicate when this is attempted. This results in a class of two features.

Altogether there are thirty-nine features. They are divided into two sets according to their "importance" in recognizing characters. The first set has thirteen features and is always used for recognizing a pattern. The features in this set are thought to be invariant over fonts and the set is referred to as the invariant set. Furthermore it is believed that they are sufficient to define most of the capital alphabetic characters. The second set of twenty-six features is used for recognizing characters only if the first set fails. The features in this set include those which may change over fonts; hence the set is called the not-so-invariant set. This set also includes the critical features.

The invariant features and their mnemonics are: straight lines - TL = top line, (horizontal line at the top), BL = bottom line, ML = middle horizontal line, LL = left line, (vertical line at the left), RL = right line, CL = center vertical line; open holes - OB = open at the

bottom, OT = open at the top, OR = open at the right, OL = open at the left; closed holes - BCH = big closed hole, CHT = (smaller) closed hole at the top, CHB = (smaller) closed hole at the bottom.

The not-so-invariant features and their mnemonics are: specific size of opening in open holes - BOB = big opening at the bottom, LOB = little opening at the bottom, BOT = big opening at the top, LOT = little opening at the top, BOR = big opening to the right, BOL = big opening to the left; more specific location of "little" open holes - LORT = little opening to the right and located at the top, LORB = little opening to the right and located at the bottom, LORM = little opening to the right and located in the middle, LOLT = little opening to the left located at the top, LOLB = little opening to the left located at the bottom, LOLM = little opening to the left located in the middle; large depth of holes from the open end to the side opposite - DHB = deep hole open at the bottom, DHT = deep hole at the top, DHR = deep hole open at the right, DHL = deep hole open at the left. The critical features are: location of greatest density, i.e., a significantly greater number of active entries in one part of a pattern-matrix than in its complement - TD = density at the top, BD = density at the bottom, ED = density at the top equal to density at the bottom, LD = density at the left, RD = density at the right,

QD = density right and left sides equal; points - IPT = interior point in a hole, XPT = exterior point to the pattern. Insurance features: indication whether a horizontal line is over half as wide as the pattern is high = WLR (wide line along a row), indication whether a vertical line is over half as wide as the pattern width = WLC (wide line along a column). See figure 9. for an illustration of how the features depict the characters.

## 3.4.4 The Representation

The coded representation of a defined character is essentially a meta-notation. Words are a notation for human speech and the letters are a notation for the sounds of human speech, they represent the sounds. Our representation, as applied to letters, (it could conceivably be applied to pictures, Kirsch, 1964, suggests such an inclusive system for the representation of patterns), is a notation which represents a description of the notations for the sounds of human speech.

The representation we use is a binary vector which has one location for each feature and a one indicates the feature is present and a zero indicates the feature is absent. For each character there are two such vectors, one represents the invariant features and the other represents the not-so-invariant features. They are: V = TL,BL,ML,RL, LL,CL,OB,OT,OR,OL,BCH,CHT,CHB and V2 = BOB,LOB,BOT,LOT,

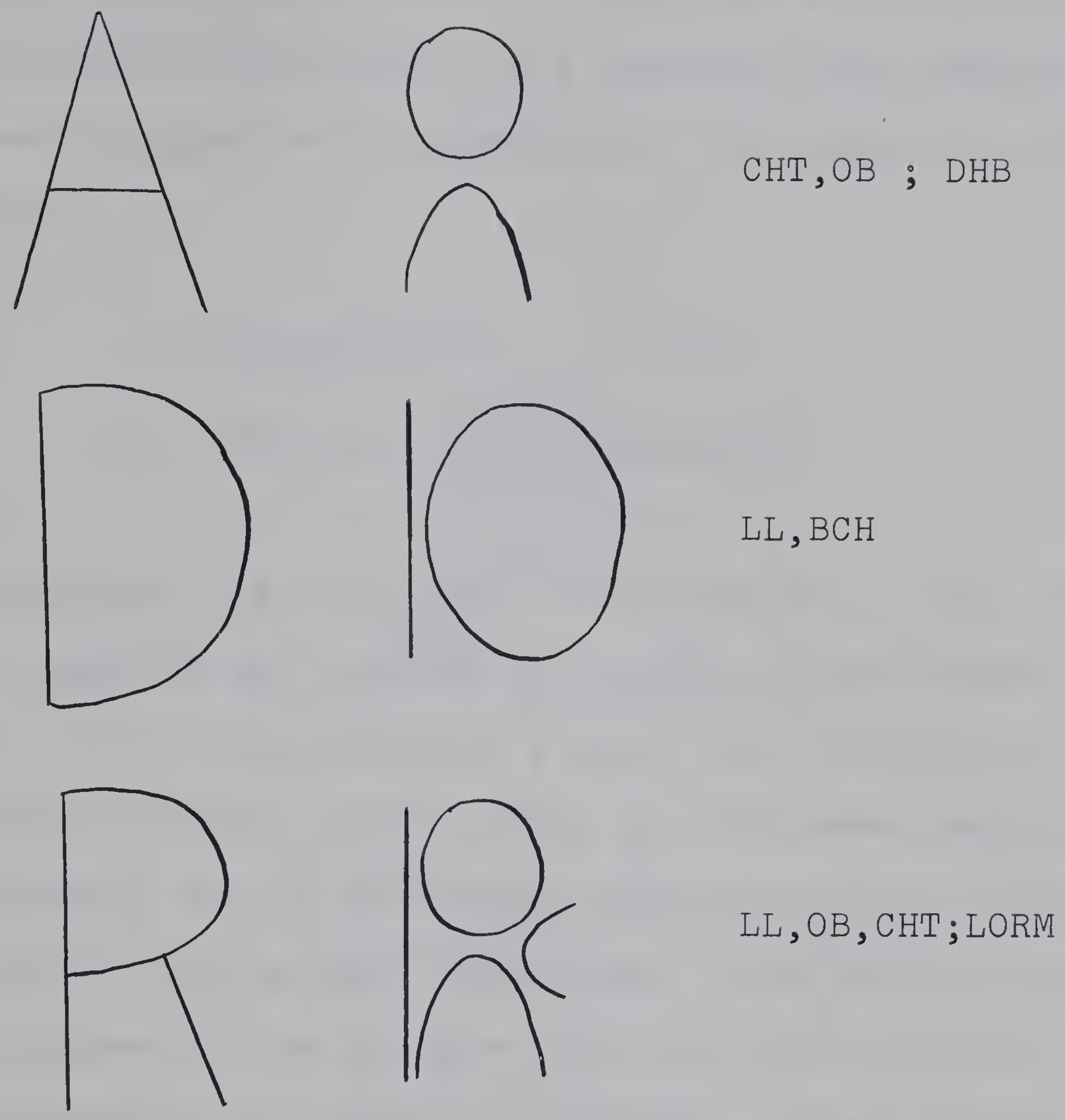


Figure 9. An illustration of how the features depict the characters using an arbitrary C-shape for open holes and on arbitrary O-shape for closed holes.

BOR,LORT,LORB,LORM,BOL,LOLT,LOLB,LOLM,XPT,IPT,DHB,DHT,DHR, DHL,TD,BD,ED,RD,LD,QD,WLC,WLR . These two vectors are the "descriptive definition" of a character when appropriate values are assigned to the mnemonics. An example is, for A we have

V = 0000001000010

V2 = 100000000000001000001100000

which represents OB,CHT and BOB,DHB,ED,RD . This representation employs the concept of duality of patterning which is: each combination of a small set of symbols represents a distinct "word"; that is, the same symbols are used repeatedly but in different locations and the location is critical. Here we use fixed length words and only two symbols whereas in the alphabet the words are variable length and there are twenty-six symbols. An example of duality of patterning - MEAT, TEAM .

Our representations for all the letters in the alphabet may be depicted graphically in several ways. One is a Venn diagram where the features are nonintersecting regions and the letters are regions intersecting the "feature" regions, see figure 10. Another way of visualizing the descriptive definition is as a vector in k-dimensional space where each axis represents a feature. For practical

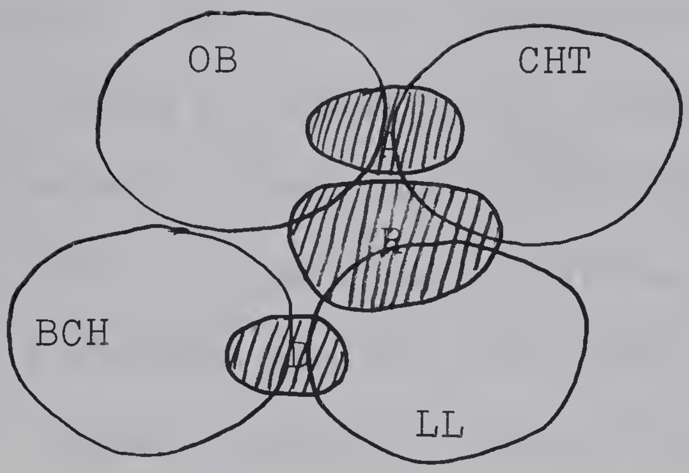


Figure 10. A Venn diagram of the features-representation of A, D, and R.

purposes a chart, such as presented in Chapter V, is used.

### 3.4.5 The Learn Mode

The learn mode of the define stage should determine the features of canonical patterns and then store this information and the pattern identification in a form suitable for future referral. The form of storage for the coded representations should be amenable to a rapid and accurate matching of subsequent coded representations.

Initially a set of operations must be programmed which will operate on a pattern-matrix and yield the desired representation. These operations should be preceded by high-level tests which decide whether a feature is absent or possibly present. These tests direct the functioning of the program to those operations which are likely to yield the features present, while bypassing the operations which cannot yield a feature present.

After the series of operations and tests are performed, the results must be collected into a coded representation and stored with its name.

A description of the learning routine and storage we used is given in Chapter IV.

### 3.4.6 The Recognize Mode

The recognize mode of the define stage should determine the features present in an unknown pattern and use this

information with the previously stored, "learned" information, to determine the names of the patterns or to indicate when a name cannot be determined. The way in which the information is used to name patterns is critical to the success and speed of the recognition procedure.

The recognize mode should use the same operations as the learn mode to determine the features present in a pattern. However since all features are not needed to distinguish some patterns from the set of patterns, the features should be grouped so that recognition can be achieved in many cases without having employed the whole set of features. In this situation not all the features-definition operations will be used unless it is necessary.

Once a sub-set of features is determined it is compared in a matching process with the stored information. If the initial sub-set of features is not successful, more features should be determined and used in a matching process. This procedure is continued until the pattern is named or all the features are determined.

There are various matching methods and they are usually dependent on the coded representation. The matching method will be described in the recognition routine in Chapter IV.

# CHAPTER IV

# THE PR SYSTEM AND THE CHAR PROGRAM

## 4.1 Introduction

In this chapter we describe the PR system, which is a simulation of an overall pattern recognition system, and includes the CHAR (mnemonic for character) pattern recognition program. The PR system provides for the input of a page of characters or a single character. If a document's page is input, the system first detects and extracts lines and then characters are extracted and either learned or recognized. The environment of the PR system and the programming language used are briefly described. Then we give an overview of the PR system which is followed by a detailed description of the CHAR program. Finally we present an evaluation of the system and suggestions for a more sophisticated learning procedure.

## 4.2 The Environment and the Programming Language

The PR system has two environments, paper and hardware. The paper environment is necessitated by the lack of peripheral light sensing equipment and it simulates the input device. The simulated input is transferred, manually, via typewriter console (IBM 2741 remote access terminal), to the hardware environment, an IBM System 360 Model 67

computer. The hardware environment contains all the programmed routines for the PR system.

The programmed routines are written in a new programming language called APL, (see Iverson, 1962, or Falkoff and Iverson, 1966). APL's set of operators are very powerful particularly with respect to its arithmetical and logical capabilities in the handling of arrays. For this reason it is particularly suited to manipulation and reduction of an input resulting in a matrix representation. As will be shown we also use this capability to program a fairly powerful and efficient "parallel" matching process in the recognize routine.

At present APL is only available as an interpreter and no object code is generated hence it is not very efficient for production runs. Furthermore APL is limited to a directly accessible storage space, called a workspace, of only 31,872 bytes which is not sufficient for our system. However, in the APL system provisions are made for communication between an "active" workspace, (the one in use), and "inactive" workspace, (those not in use but stored in the computer). For the simulated operation of the PR system we used several workspaces and transferred the results from the previous active workspace to the active workspace in an organized sequence of activating and storing workspaces.

## 4.3 The PR System

The PR system is a simulation of a pattern recognition system. It includes the hand simulation of an input device containing a hexagonal element sensing grid, the mechanical fovea, as described in Section 3.1.4 of Chapter III. The details for the hand simulation of input are given in Appendix A. The mechanized portion of the PR system is best described in terms of the functions of its workspaces which are named STORE, ONE, TWO and CHAR. STORE receives the input data, ONE extracts line matrices from the input "page" matrix, TWO extracts character matrices from a line matrix and CHAR operates on the character matrix to obtain its linear representation and either learns or recognizes the pattern. For a flow diagram of the PR system, in terms of its workspaces, see Appendix B.

STORE contains a 4 × 20 matrix of which the elements are 25 × 40 pattern matrices, a 100 × 800 "page" matrix and a function for transferring the pattern matrices into the page matrix. (NOTE: in APL an independent program module or subprogram is called a function, this is similar to the function subprogram and the subroutine of FORTRAN.) This allows for the storage of 80 individual patterns <u>and</u> a page approximately four lines of twenty characters each, i.e., another 80 or more patterns.

The workspace ONE contains instructions for obtaining

input matrices from STORE and initiating the PR system routines, in the function GETDATA. A function for initiating the page manipulation routines, called BEGIN, requests the user to specify whether the data is numeric or alphanumeric, whether to learn or recognize the input data and if recognizing data whether intermediate "recognized" results are to be shown before the recognition of all the characters on the page is completed. BEGIN initializes the page manipulation variables, extracts the top line matrix, and transfers control to the pattern extraction function, BEGIN2, in workspace TWO. Since the transfer of control to a different workspace must be done manually, we indicate a transfer with a display of the instructions for affecting the transfer. A function called GO extracts all the subsequent line matrices in a page matrix - transferring control to BEGIN2 of workspace TWO after each line matrix extraction - indicates when all lines are processed and, if the system is in the recognize mode, displays the complete results of the recognition of a page of characters.

The workspace TWO contains a function called BEGIN2 which is used each time a new line is started. This function initializes the line matrix manipulation variables, extracts the leftmost character matrix in a line matrix and transfers control to GO in the CHAR workspace. All

subsequent character matrix extractions and transfers -of-control (to GO in the CHAR workspace after each extraction) are done by the function GO2. GO2 also indicates spaces between sequences of characters, i.e., to separate words. Finally, GO2 indicates the end of a line matrix processing and transfers control to GO in the ONE workspace to obtain another line matrix. Both BEGIN2 and GO2 request the name of the pattern matrix extracted, if the system is in learn mode.

The CHAR workspace contains functions for initializing the definition and position parameters, SETPRS, and the data collection vectors, ICPR. These functions are used at the beginning of an experiment. The other functions in CHAR are nested in a function called GO. This function accepts data from workspace TWO, normalizes a pattern matrix as described in Section 3.3 of Chapter III (normalizer yields the pattern matrix to be operated upon, and its height and width), determines the features present and either stores the representation (in learn mode) or compares the representation to the stored representations (in the recognize mode). When a pattern matrix has been processed, GO transfers control to GO2 in the TWO workspace to obtain another pattern matrix. Several of the feature-determining functions contain a means for collecting data on the features. The functions in GO will

be discussed in further detail later in this chapter.

The operation of the PR system is as follows. The input is manually coded (see Appendix A) and entered via typewriter console remote terminal into the STORE workspace which is activated by typing )LOAD STORE. This workspace is deactivated and the input retained by typing )SAVE STORE. Workspace ONE is activated by typing )LOAD ONE, and instructions for initiating the mechanized portion of the PR system are obtained by typing GETDATA. From this point in the system, the operation is self explanatory. The functions in all the workspaces request necessary information and give instructions for the activation and deactivation of the workspaces in the appropriate places.

## 4.4 The CHAR Program

### 4.4.1 Introduction and Definition Parameters

The CHAR program is the chief module of the PR system. It contains the set of functions for operating on a pattern matrix (to define the pattern in terms of features), the functions for learning, the learned representations, and the function for recognizing a pattern. These functions are stored in the CHAR workspace, most of which are nested in the function GO. For a flow diagram of the CHAR program see Appendix B.

Before we describe the functions in detail we will

define the way we measure features. Each pattern to be learned or recognized may result in a different size pattern matrix. The width, PW, is determined when the pattern matrix is extracted from the line matrix and the height, PH, is verified or redetermined before the pattern matrix is "filled-in" in the normalization procedure. (The height is originally taken as the number of rows in the current line matrix when it is extracted from the page matrix, and it is redetermined only if there is an indication that the original PH is too big.) All of the features in a pattern are determined with respect to the pattern matrix width and height.

Definition parameters are used to determine the percentage of the pattern matrix width or height that a feature indication must be in order to be a valid feature and to determine the feature's position. An example is: Suppose the line definition parameter, LDP, is .77 and the line position parameter for top lines, LPP, is .2 and a pattern matrix width and height were found to be 16 and 20 respectively, i.e., $PW = 16$ and $PH = 20$. When testing for horizontal lines we find a maximum row sum to be 13, hence we have $13 \div 16 > .77$ and the row is considered as a line. (A further verifying test is made as described in the operational description of the function LINES.) The position of the row in the pattern matrix is

[illegible]

determined as, say, 2 and we have $2 \div 20 < .2$ hence the proposed line's position is at the top. That is, when a row or column sum divided by the pattern width or height, respectively, is greater than or equal to the line definition parameter a line is tentatively defined; its position depends on whether it is a row or column and within 20% of the top or left side of the pattern matrix, within 20% of the bottom or right side or neither.

The definition parameters are size bounds for line length, hole dimensions, hole opening width and hole depth, and position bounds for lines and holes, e.g., line definition parameter = LDP = 0.77, big hole definition parameter (for dimensions and opening) = BHDP = 0.5, little deep definition parameter (for hole depth) = LDDP = 0.12, line position parameter = LPP = 0.2, open hole position parameter bottom (relative location of an opening on the side) = OHPPB = $2 \div 3$. For the complete set of definition parameters see SETPRS in Appendix B. The orientations for lines and for the opening of holes are determined from the way a test is made.

At the present time the programmer provides the values for the definition parameters. A proposed method of mechanically determining the initial parameter values from a set of patterns and/or subsequently altering the initial values depending on results obtained by the CHAR

program, is presented in Section 4.5. The function SETPRS contains all the definition parameters with their values which are set during the first run by typing SETPRS.

### 4.4.2 <u>The Normalize Routine</u>

The GO function first re-establishes the names of some variables which are transferred from workspace TWO in the form of a vector, i.e., instead of transferring each variable we specify a vector of the variables and transfer only the vector. Then it uses the function FILLIPM to fill in the gaps of the pattern matrix; this was described as the normalize stage in Section 3.3 of Chapter III. FILLIPM uses the function FIXPH to re-determine the pattern height if necessary.

We shall begin an example here which will be used to illustrate this routine and subsequent routines. Suppose a page of patterns, where the left-most character in the first line is an A, has been entered and saved in the 100 × 800 matrix, M, in workspace STORE. We save STORE, activate workspace ONE and type GETDATA which tells us how to provide space in workspace ONE for M and how to transfer M from STORE. After having completed the transfer and initializing the <u>i</u>nput <u>p</u>age <u>m</u>atrix (IPGM) to M we begin execution of the PR system by typing BEGIN. A message instructs us to specify MODE by typing 0 or 1, 0 for learn, 1 for recognize. We type 0 (learn).

Another message asks us to signify whether the patterns are of alphanumeric or only numeric characters. The first line (LINE) is extracted from the page and instructions are given for transferring control to workspace TWO. After completing the transfer (i.e., saving ONE, activating TWO, and copying LINE, MODE and other parameters as indicated), we type BEGIN2 which, since it is in learn mode, requests the name of the first pattern to be encountered in the line, extracts the left-most pattern from the line and gives instructions for the transferring of control to workspace CHAR. We affect this transfer by saving TWO, activating CHAR, copying the input pattern matrix (IPM) and other values such as mode, line height, pattern width, etc.; then, assuming the definition parameters have been previously initialized, we type GO. (If the definition parameters had not been initialized we would have typed SETPRS before typing GO.)

The first four instructions of the function GO are the normalize routine. Instructions [1], [2], and [3] re-establish the alphanumeric or numeric designation (ALNUM), the parameter used in the recognize mode to show intermediate answers (SEE), and the parameter which indicates a large space between patterns, PGS, from a vector that was transferred from workspace TWO. Instruction [4] calls the function FILLIPM which re-esta-

blishes the transferred line height and pattern width, then verifies that the pattern height is the same as the line height. (If the line height is not the same as the pattern height, FIXPH is called to determine the pattern height. We shall assume line height equals pattern height in our example.) The pattern height and pattern width are PH = 18 and PW = 27 respectively. FILLIPM then replaces every zero with a one on either side of it by a one, first for rows then for columns. Figure 11. shows the pattern, IPM, transferred from workspace TWO and Figure 12. shows the "filled" IPM.

### 4.4.3 The Define Routine

After the normalize-routine function has operated on the initial pattern matrix, the function GO enters the define routine which uses functions to determine the features by operating on the resultant pattern matrix. First the features are initialized at the value zero by the function SETCAT. Then the function LINES determines all the line features in a pattern, i.e., if a top line is present TL = 1 . Next OHOLES indicates the open holes present. Then OHD determines the depth of the indicated open holes and firmly establishes the open holes. CHOLES follows and indicates the closed holes in the pattern. CKCH then checks the indicated closed holes to ascertain that they are closed. At this point a

```
      IPM

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0
0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0
0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1
```

Figure 11. An example of a pattern matrix as it appears when transferred from workspace TWO.

```
        IPM


   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0
   0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
   0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
   0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0
   0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0
   0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0
   0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0
   0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0
   1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1


        PH,PW
 18   27
        NAME
 A
```

Figure 12. The same pattern matrix as in figure 11. after it has been normalized.

branch to the recognize routine is made if the program is in recognize mode. If the program is in learn mode the critical feature classes termed points and density are determined by the functions XTNS and DEN, respectively. The program then enters the learn routine. The learn and recognize routines will be discussed in the following sections. Below, we describe some of the key points of the functions in the define routine.

LINES first calculates the row and column sums, then obtains information about these sums in the function INFO. A test is made to determine if any lines are indicated. That is, if there is no value of a row or column sum divided by PW or PH, respectively, that exceeds the line definition parameter, no lines can be present and the rest of the LINES function is skipped. If lines can be present, tests are made employing function LIN, with the appropriate variables each time, to determine the orientation and position of the indicated lines. The indicated lines are further tested to ensure there are no breaks in the line. If there are no breaks the value 1 is given to such lines, all other line values remain zero. The critical features WLR and WLC are also determined by LINES. WLR has the value 1 if the number of line-indication row sums divided by the pattern height is greater than or equal to LWMP = .55 , where LWMP is

the line width (too) much parameter. This also causes a branch around the horizontal line determination operations. WLC is the corresponding variable for columns. LINES yields the value one for LT, BL, ML, LL, RL, CL, WLC and WLR when any of the corresponding features is established, otherwise these features remain at zero as initialized by SETCAT.

Continuing our example, when the define routine is entered SETCAT initializes all feature values to zero, then LINES operates on IPM. LINES obtains the row and column sums then uses the function INFO to determine whether any column or row sum exceeds or equals the line definition parameter. Since the maximum column sum is 8, and $8 \div 18 < .77$ $(18 = PH)$ and the maximum row sum is 12, and $12 \div 27 < .77$ $(27 = PW)$, there can be no lines present, so TL, BL, ML, RL, LL, CL all remain zero. WLC and WLR also remain zero since there are no lines, so there can be no lines that are more than half as wide as the pattern width or height.

Returning to the description of the define routine, OHOLES begins with the bottom row of the pattern matrix and determines if there is a string of ones and zeros such that the number of consecutive zeros, bounded on each end by a one, divided by the pattern width exceeds or equals the big hole definition parameter or, if not, the little

hole definition parameter. For example if the binary string is 110000010, PW = 9, and BHDP = 0.5 then E = 8, B = 3 and BOB = 1 where BOB ← BHDP ≤ (E-B) ÷ PW . OHOLES uses the function OH to perform this latter procedure for a general case. OH employs the function LG10 to locate the beginning, B, of the first bounded sequence of zeros and the beginning of the second sequence of ones, E. OHOLES performs the same tests for the top row, left-most column and right most column using OH with the appropriate variables. The procedure is repeated for the next row from the top, from the bottom etc., until the outer quarter of the pattern matrix has been analyzed or until an "open" hole is found on each side of the matrix. Whenever an open hole is indicated on a side of the matrix the procedure for that side is skipped for the remainder of the test cycles. The function OTBM is used, when right or left little open holes are indicated, to determine whether they are near the top, bottom or middle of the side indicated. The function OHOLES may yield one values for BOB,LOB,BOT,LOT,BOR,LORT,LORB,LORM,BOL,LOLT, LOLB,LOLM when any of the corresponding features is indicated, otherwise these features remain at a zero value as initialized by SETCAT.

In our example when OHOLES tests the bottom row of the pattern, it finds a sequence of ones, zeros, ones

where the number of "enclosed" zeros is 17. Since $17 \div 27 = .629629$ (PW = 27) is greater than the big hole opening definition parameter, (.5), BOB equals 1. When the top row left-most and right-most columns are tested, no sequences of zeros enclosed by ones are found. The tests begin again and since BOB = 1 no test is made at the bottom; however, the row second from the top row and the second column from the left and right are tested, etc. This procedure continues until five rows (columns) from the top, left and right sides have been tested. OHOLES results in BOB = 1 , and BOT,BOR,BOL,LOB,LOT,LORM, LORT,LORB,LOLM,LOLT,LOLB all equal to zero.

OHD is skipped if no open holes have been indicated by OHOLES. If open holes have been indicated, a center column was determined in OHOLES, i.e., if BOB = 1 then open bottom middle, $OBM \leftarrow B + \lfloor (E-B) \div 2$ , was calculated, where $\lfloor$ means truncate the decimal part of the value $((E-B) \div 2)$ . OBM is used in OHD to select the center column and then OHD determines the number of zeros before the first one (1) in the column starting from the direction of the opening. If the opening is to the right or left, the number of zeros is divided by PW; if it is to the top or bottom the number is divided by PH. This yields a proportional depth for each open hole. If the proportional depth does not exceed the <u>l</u>ittle <u>d</u>eep (hole)

definition parameter, LDDP, the hole is declared not present. Otherwise it remains as defined by OHOLES. If the center column, or some column near the center column, has a proportional depth which exceeds the big deep (hole) definition parameter, BDDP, a deep hole is declared. This procedure is performed for all sizes, orientations and positions of open holes found by OHOLES. Thus OHD confirms the open holes features indicated by OHOLES and indicates "extra" hole depth by setting DHB,DHT,DHR and DHL equal to one.

In our example when OHD is called it uses OBM = 22 , which was determined in OHOLES, to select column 22 and it determines the location of the ones closest to the bottom row of the pattern matrix. The ones start 10 rows from the bottom and since B = 10 ÷ 18 = .555 , which is greater than .2 (the little-deep definition parameter), BOB remains one. Several columns on either side of the center column are tested simultaneously and their number of zeros from-the-bottom are divided by PH = 18 . If at least one of the results, say B, is greater than .25 (big-deep definition parameter), DHB = 1 . Since B > .25 DHB = 1 . OHD then determines which other open holes have been indicated by OHOLES and, since there are none, it terminates. OHD results in retaining BOB = 1 and in setting DHB = 1 . DHL,

DHR and DHT all remain zero.

CHOLES skips the test for big closed holes if DHR or DHL = 1 . In the big closed hole test a column and a row, that intersect at or near the center of the pattern matrix, are inspected by the function CH to determine whether both contain a sequence of zeros, bounded by ones, such that the number of zeros in the sequence divided by PH for the column and PW for the row exceeds the big hole definition parameter, BHDP. If the test on the row fails, the test on the column is not made. If both tests succeed a big closed hole is indicated, BCH = 1 , and no other closed holes are sought. If both tests fail CHOLES determines whether the two other closed holes are possible; that is, if $(BOT \vee LOT) \wedge (BOB \vee LOB) = 1$ it is postulated that the closed hole top and the closed hole bottom are not possible and no more tests for closed holes are made. If either of these closed holes is possible, a test is made to determine whether a closed hole at the top is possible. If this hole is not possible, the function begins testing for closed holes in the bottom portion of the pattern matrix. The tests for the smaller closed holes, using the function CH, are much the same as those for the big closed hole. However CH operates on any row containing zeros enclosed by ones and on all the columns which intersect the interior zeros

of the row. The selection of rows continues until a hole is determined or the appropriately positioned rows are exhausted. The function CHOLES gives one values to BCH, CHT,CHB when any of the corresponding features is indicated, otherwise these features remain zero.

Our example will illustrate how CHOLES is used. When CHOLES is entered with DHR and DHL both zero, it tests for a big closed hole and finds none. Since $(BOT \vee LOT) \wedge (BOB \wedge LOB)$ is zero, CHOLES continues by testing for a closed hole at the top, It begins testing for a sequence of ones in a row that is not equal to the row sum. When such a sequence is found in row 4, a test is made for a sequence of ones, zeros, ones in that row. Such a sequence is found and the number of enclosed zeros, 2, is divided by $PW = 27$. But $2 \div 27 \not\geq .12$, where .12 is the little hole definition parameter (LHDP), so CHOLES resumes testing for a sequence of ones in a row that is not equal to the row sum, etc. Finally a sequence of ones, zeros, ones is found in row 7 such that the number of enclosed zeros, 4, divided by PW exceeds LHDP, i.e., $4 \div 27 \geq .12$, and a horizontal hole HH is declared. Then a search is made on the columns intersecting the "enclosed" zeros of row seven. This search is made using the test for ones, zeros, ones such that the number of enclosed zeros divided by the pattern

height (PH) is greater than or equal to LHDP. The required column, 14, is found on the second try, i.e., intersecting the second enclosed zero of row seven. The column, 14, has 5 enclosed zeros and since $5 \div 18 \geq .12$ (PH = 18), a vertical hole is declared, VH = 1. Now since $VH \wedge HH = 1$ we have a closed hole at the top, CHT = 1. A test is made to determine if a closed at the bottom is possible and since $BOB \vee LORB \vee LOLB = 1$, CHB is not possible and CHOLES terminates. CHOLES results in CHB and BCH equal to zero and CHT equal to one.

CKCH is a check to verify BCH,CHT,CHB and is skipped if none of these has been indicated by CHOLES. The function CKCH checks that there are no gaps in the sides of a closed hole. The portion of the columns and rows from the hole center to the outside of the hole contour are tested by first selecting those columns on one side of the center (including the center column if there is an uneven number of columns) and doing an "or" along each half row to and including the column on the furthermost edge of the pattern matrix. This "or" operation along the half rows is done simultaneously across all the columns selected and it produces a vector. If the beginning of the second sequence of zeros in this vector do not exceed the beginning of the second sequence of ones in the center column, the closed hole is given a zero value. A

similar test is made for each of the four directions, where rows are selected from the center out for two directions, and columns for two. Each closed hole, that was indicated by CHOLES, is checked in this manner. The function CKCH yields one values when the indicated closed hole features are verified, otherwise these features have a zero value.

To illustrate how CKCH works we refer to our example. CKCH uses the location of the closed hole at top, that is the intersection of the VH and HH which is element (7,14) in the matrix, as a center point from which to check for gaps in the contour surrounding the hole. The first test "ors" along the rows from column 1 through column 14. The result is a vector; this vector is examined and the beginning of its second sequence of zeros noted as 19. Since 19 exceeds the beginning (9) of the second sequence of ones in the "center" column, the left side of the hole is declared free of gaps. The same procedure is repeated for all sides of the closed hole and no gaps are encountered so CHT remains one. CKCH terminates after ascertaining that no other closed holes have been indicated.

The function XTNS is used only if at least one of BCH,BOR or LOR has a value one. XTNS sums the pattern matrix columns and determines whether there is a column sum less than three for the rightmost three columns. If

there is such a sum an exterior point is indicated. XTNS uses the function CIP to determine whether there exists a sequence such that ones occur, then zeros, then ones, then zeros, then ones in any column in the region from about one-third of the way from the left side of the pattern matrix to about three-fourths of the way. If there is such a sequence in any of these columns an interior point is indicated. The features resulting from XTNS are mainly to distinguish the character G from C and Q from O. The function XTNS yields one values for XPT and IPT when the corresponding features are established, otherwise these features remain at a zero value. In our example when XTNS is entered it finds that none of BCH, BOR or LOR is one so it terminates.

DEN (for density) divides the matrix at its center column forming two sets of columns (if there is an uneven number of columns, the center column is included in each of the two sets). The sum of the column sums is determined for each set. One such sum is subtracted from the other and if the difference is greater than .05 times the number of ones in the pattern matrix, the side corresponding to the minuend is given a density indication. The subtraction is repeated reversing the sums etc. Finally the absolute value of the difference is taken and if it is less than or equal to the tolerance mentioned

above, equal density is indicated. For example if (sum of "left" column sums) - (sum of right column sums) > tolerance, LD = 1 . The same procedure is done for rows. The function DEN yields one values for LD,RD,QD,TD,BD,ED when the corresponding densities are established, otherwise these features remain at a zero value.

Continuing the example, when DEN is entered the tolerance is calculated as the intregral part of five percent of the total number of ones in the pattern matrix, i.e., $.05 \times 106 = 5.3$ (where 106 is the total number of ones), and the tolerance is 5. The set of column sums for columns one through fourteen is summed and this sum is 41; then the sum for the set of column sums for columns fourteen through twenty-seven is determined as 67. Since $67 - 41 > 5$ , right density is declared (RD = 1); LD and QD remain zero. Then the set of row sums from row one through row nine is summed and the row sums for rows ten through eighteen are summed. Both of these latter sums are 53 and we have $|53-53| < 5$ , hence equal density for top and bottom is declared (ED = 1) while TD and BD remain zero.

The features OB,OT,OR,OL are composite features and are determined just before storage in the learn routine or just before recognition in the recognize routine. Open bottom, OB, is one if DHB is one and BOB or LOB is

one. OT,OR and OL are determined similarly using the values of the corresponding features in the appropriate orientation.

### 4.4.4 The Learn Routine

If the program is in learn mode, GO enters into the learn routine after completing all the functions in the define routine. The learn routine begins by giving the location of the previously learned pattern information. This location is called PN for pattern number. It then asks for a new pattern number which designates the storage location for the current pattern's representations and name. The function CAT1 is used next; it determines the invariant vector representation and stores it and the pattern name. Then the function CAT2 is used to determine and store the not-so-invariant vector representation. Finally the function LCLNUP transfers control to the function, GO2, in the PR system which extracts a pattern matrix from the line matrix.

When the learn routine is entered each feature, except the composite features, has been defined. CAT1 determines the values for the composite features and then each feature mnemonic has a significantly assigned value of zero or one. These values for the feature mnemonics determine the vector representations. The function CAT1 determines the invariant vector, V, and

stores it as a row in the invariant category matrix C1 or C11, where C1 is used for the storage of alphanumeric patterns' representations and C11 is used for the storage of only numeric patterns' representations. (Some representations for numerals can not be stored with the representations for capital letters due to their identity; however, in any case where confusion will not arise, the representations for both will be stored together in C1. That is C1 will contain the representations for the capital letters and some numerals while C11 will contain only the representation for all the numerals.) The pattern number, PN, specifies the row that V is to occupy in C1 or C11. The name or symbolic expression of the pattern learned is entered into the vector of names, N1, in the location specified by PN.

The function CAT2 determines the not-so-invariant vector, V2, and stores it as a row in the not-so-invariant category matrix C2 or C22, where C2 is used for the storage of alphanumeric patterns' representations and C22 is used for the storage of only numeric patterns' representations. The row where V2 is to be stored is specified by PN. The function LCLNUP gives instructions for the transfer of control to GO2 in workspace TWO where another pattern matrix to be "learned" is obtained.

When the PR system is in the learn mode a set of

canonical pattern matrix representations is determined and stored. The resultant category matrices C1,C11,C2, and C22 will contain these representations as rows in their matrix structure.

We use our example to illustrate the learn routine. When the learn routine is entered the pattern number (PN) of the previously learned representation is displayed as 0; then an instruction is displayed which requests the operator to type the pattern number for the present representation so we type 1. Next, the function CAT1 is called and in CAT1 the feature OB = DHB ∧ (BOB ∧ LOB) becomes one since BOB and DHB are both one. The feature OT,OR, and OL remain zero since their composite values are zero. Now all the features have a significant value (the features with value one are BOB,DHB,CHT,RD,ED and OB while all other features have value zero) and the invariant vector representation V is determined from V = TL,BL,ML,LL,RL,CL,OB,OT,OR,OL,BCH,CHT,CHB ; i.e., V = 0000001000010 . Then V is stored as row 1 (PN) in the invariant category matrix C1 (in APL notation this is C1[PN;] ← V) . The name (NAME), which had been specified in workspace TWO before the pattern matrix was extracted from the line matrix and which had been transferred to workspace CHAR, is now stored in the name vector (N1) as the first symbol (N1[PN] ← NAME) . Next

the function CAT2 is entered and the not-so-invariant vector representation V2 is determined from V2 = BOB,LOB, BOT,LOT,BOR,LORT,LORB,LORM,BOL,LOLT,LOLB,LOLM,XPT,IPT,DHB, DHT,DHR,DHL,TD,BD,ED,RD,LD,QD,WLC,WLR; i.e., V2 = 1000 0000000000100000110000 . The vector V2 is stored as row one (PN) of the not-so-invariant category matrix C2. If the example pattern had been of a numeric character the vector representations V and V2 would have been stored in the category matrices C11 and C22 respectively, and if advisable, later stored by the operator in C1 and C2. After the vector representations have been determined and stored the function LCLNUP is called. This function gives the transfer of control instructions which when implemented saves workspace CHAR, activates workspace TWO and extracts the next pattern matrix from the line matrix.

### 4.4.5 <u>The Recognize Routine</u>

If the program is in the recognize mode, GO enters into the recognize routine after passing through some of the functions of the define routine. The recognize routine begins by determining whether the patterns to be recognized have been specified numeric or alphanumeric. If they are numeric the category matrices C11 and C22 are used in the function REC. If they are alphanumeric the function REC uses the category matrices C1 and C2. The function REC is then used to determine the name of the pattern. The

last function in the recognize routine, RCLNUP, is then employed; it provides for intermediate display of results and storage of results in a line format, and transfers control to the function in the PR system which extracts a pattern matrix from a line matrix.

When the recognize routine is entered several - but not all-features have been defined. The function REC determines values for the composite features and then each feature mnemonic in the invariant set of features has a significantly assigned value of zero or one. REC then determines the invariant vector, V, and uses it in the first "matching" method employed to recognize the pattern.

REC has several methods for recognizing a pattern where each subsequent method is used only if the former method fails to give exactly one name. These methods in the order of their use are:

1) The vector V selects the columns of the invariant category matrix, say C1, which correspond to the ones in V. This results in a sub-matrix of as many columns as there are ones in V and the same number of rows as in C1. (The operation in APL is V/[2]C1.) Each row in the sub-matrix is operated on by the "and" logical operator. This yields a vector which is used to select the name from the name vector, N1. If exactly

one name is selected the pattern is recognized as that name and the REC function terminates.

2) The second method begins by calling the functions XTNS and DEN which operate on the pattern matrix to define the remaining undefined not-so-invariant features. Now all the features have been defined and have a significant binary value. The not-so-invariant vector, V2, is determined and used to select columns from C2. The vector resulting from an "and" across these columns is used to select the name or names. If exactly one name of this (name) result is contained in the (name) result of method 1. it is taken as the desired name.

3) The third alternative method sums the rows of the sub-matrix, which was extracted by the vector V2 from matrix C2 in the second method, and the row number(s) with the maximum sum is used to select a name or names from N1. If exactly one of these name(s) is in the name result from method 1. it is the answer.

4) The fourth method sums the rows of the sub-matrix resulting in method 1. and the row number(s) with the maximum sum is used to select a name(s) from N1. If only one name results, it is the answer.

5) The fifth method "intersects" the name(s), determined by the row sums of C2 in the third method,

with the name(s) determined by the row sums in the fourth method in an attempt to obtain a single name answer.

6) The last method intersects the name(s) determined by the "and" operation over the selected columns of C2 (performed in the second method) with the name(s) selected by the row sums in method 4.

If none of the methods yield a single name the pattern is declared unknown by being named ? .

In the first method we assume the features are invariant; the second method allows for some noise and for critical features. The methods three through six are experimental allowances for noise. In most cases, thus far encountered, methods one and two are sufficient. When each different method is used, i.e., when the not-so-invariant vector is first used etc., the function records this fact as well as the results from the selections and intersections. These records could be used to indicate the degree of certainty of the answer. In its present state the program gives no such indication.

Although the recognition procedure is sequential in its choice of methods used to determine a pattern name, each method is essentially parallel in that a set of features for the unknown pattern is determined and used simultaneously to identify the pattern. If the methods

were sequential, an attempt to identify the pattern would be made after each feature is evaluated. This would give much more weight, implicitly, to some features than to others and none could have the same weight.

The function RCLNUP displays the resulting name from the function REC, if it is desired. RCLNUP also stores a line of results of recognition in a line of a page when twenty-five characters have been recognized, and reinitializes the line. Finally, RCLNUP gives instructions for the transfer of control to GO2 in workspace Two where another pattern matrix to be recognized is extracted from the line matrix.

When the PR system is in the recognize mode a vector representation of a pattern is compared simultaneously with all the learned vector representations in a category matrix and the pattern is identified.

We will use the example to demonstrate the recognize routine, assuming that MODE had been specified one (recognize). Further we assume that three alphanumeric patterns have been learned and C1 is

```
0 0 0 0 0 0 1 0 0 0 0 1 0
1 1 0 1 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 . . .                0
.                        .
.                        .
.                        .
0 0 . . .                0
```

C2 is

```
1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 . . .                                           0
.                                                   .
.                                                   .
.                                                   .
0 0 . . .                                           0
```

and N1 is 'A', 'B', 'C', ' ', ..., ' '. That is, the first three rows of each category matrix are representations of patterns and all remaining rows contain only zeros which have no significance. (In APL matrices must be pre-structured to enable reference to a single row or column; C1 and C2 are large enough for 40 representations using 13 and 26 features respectively, i.e., C1 is a 40 × 13 matrix and C2 is a 40 × 26 matrix.) The first three elements of N1 are the names corresponding to the first three rows of C1 and C2. Provision has been made for 40 names.

Since MODE = 1 the function GO branches to the recognize routine before the call for the function XTNS and DEN has been encountered, hence these two functions have not been executed. All of the invariant features have been determined except for the composite features but some of the not-so-invariant features do not have significant values. For the example pattern matrix, the features with value one at this point in the program

execution are BOB,DHB, and CHT.

Upon entering the recognize routine it is determined that the character to be recognized has been specified as belonging to the alphanumeric set by noting that ALNUM = 0 . Hence C1 and C2 are used in the function REC rather than the category matrices containing only numeric patterns' representations (C11 and C22). The function REC is called and it determines the composite features OL,OR,OT and OB where all are zero except OB, i.e., $OB = DHB \wedge (BOB \vee LOB) = 1 \wedge (1 \vee 0)$ . Now all of the invariant features have been defined (CHT and OB equal one, all others are zero) and REC evaluates the invariant vector V = TL,BL,ML,LL,RL,CL, OB,OT,OR,OL,BCH,CHT,CHB , i.e., V = 000000100001 0 . The vector V is then used to select the columns from C1 corresponding to the features in the unknown pattern. That is columns 7 and 12 are selected and we have the 40 × 2 matrix

$$\begin{array}{cc} 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 0 & 0 \end{array}$$

Each row of this matrix is operated upon by the "and" operator which results in the vector 1000...0 . This vector selects the name, A, from N1 as the "answer" (ANS).

(In APL the operations from selecting the sub-matrix to this point are written as one statement, ANS ← (∧/V/[2] C1)/N1) . Since the answer is a single name recognition has been achieved so REC uses the parameter PGS (pattern group spaces) to select a space (S), if PGS is 1 (we will assume PGS = 0 , hence S is empty rather than a space). Then S and ANS are concatenated to the previously recognized patterns (OUT).

The function RCLNUP checks that the output line is less than 25 characters and hence need not be stored as a line in a page of output. Finally RCLNUP gives instructions which when executed saves workspace CHAR, activates workspace TWO and initiates the extraction of another pattern matrix from the line matrix.

## 4.5 Evaluation of the PR System

The chief module of the PR system, the CHAR program does not depend on details of the pattern matrix to such an extent as does Grimsdale's program. CHAR does not depend on contour or registered line width. Specific matrix configurations are not inspected piece-wise along the contour but rather as a whole in search of attributes of hypothetical configurations. Furthermore an attribute may have a range of values, proportional to the pattern matrix, and still be detected. The program does not depend on the character size except to the extent that sufficient

information is critical, hence a redundancy of information, as is given by the h-element grid coding, is desired.

One limitation of the CHAR program is its inability to recognize the straight lines if they are slanted too much. If the slant is not too great and/or there is considerable redundant information this limitation will not hinder effectivness.

The learning routine of the CHAR program is very elementary and is essentially a method of storage. There are two ways more sophisticated learning could be implemented. One is a program to determine the (initial) values of the definition parameters. Such a program would be given the features present in pattern matrices and collect data on them to determine the definition parameters. That is, say, a set of pattern matrices is operated upon and for each of these a line is given in a particular orientation and position. Say one such pattern matrix is given a LL present, then the maximum column sum in the left 20% of the pattern-matrix is divided by the PH and saved. (If the pattern matrix is given only a LL present, then the maximum column sum divided by PH and its row location divided by PW would both be saved, the latter to determine a position definition parameter.) This ratio along with similar ratios for lines is saved in one of two ways, either only the minimum ratio is saved

each time or all of them are saved as separate values. After many pattern matrices with lines have been operated upon, the value taken as the definition parameter for lines will be given by (a) the approximate mode of the saved ratios or (b) the final minimum, or some value close to (a) or (b) and perhaps on the lower side. An average will not give good results as it will tend to "lose" many lines if used as a definition parameter.

Three drawbacks to this procedure are (1) a large number of pattern matrices need to be operated upon for fairly reliable definition parameters, (some features occur in very few patterns), (2) the dependence of the definition parameters upon the set of pattern matrices used for determining them, and (3) the difficulty of predicting all the features in a pattern, (some features occurred in the representation of those patterns learned which were not noticed before). The first objection involves a lot of machine time for what may yield no better definition parameters than those chosen by the programmer. The second objection could be an advantage. If one knows what fonts are to be used for the program's subsequent production runs, the definition parameters could be tailor-made for these fonts for improved performance. However if one does not have this control over the fonts, an attempt must be made to include all

possible fonts in order to achieve general definition parameters. The implementation of the program described above was started but not completed.

The second way for more sophisticated learning is to collect data in the CHAR program on those features determined, i.e., their size and position ratios, and use this data to change the definition parameters. Here again an average should not be used. But the minimum ratio averaged with the current definition parameter and a function used on the result, which causes the definition parameter value to change much faster in one direction than in the other, e.g., for the line definition parameters, large values decrease more rapidly than smaller ones. Other suggestions are as in the discussion for determining initial definition parameters where the current definition parameter is included before a mode or a minimum is determined. The definition parameter should not be changed after each pattern matrix is processed, unless a feedback technique is used to assure the new definition parameter gives the same desired results as the old definition parameter. It is believed that the changing of the definition parameters will be more effective when data from several pattern matrices is available. However in this case a feedback should be effected on all the pattern matrices from which the data was collected to assure

"good" changes in the definition parameters.

Some mechanics exist in the CHAR to collect data that can be used to change the definition parameters as described above. This learning procedure was attempted but the results were erratic. The definition parameters changed dramatically and gave terrible feedback results. The feedback used was just a rerun of the pattern matrices to see if the new definition parameters worked. More work is needed in this area to develop an effective feedback routine which adjusts the definition parameters.

When this learning procedure is operational it should make the program even more flexible with regard to fonts. However in productive operation it is believed that any learning and feedback should be deleted for the major portion of a document as it would hinder the speed considerably. This attempt for a more sophisticated learning procedure increases the pattern processing time somewhat since there are many data collection statements that must be interpreted each time since they remain in the program.

The PR system is an experimental model for a proposed pattern recognition system. In its present state it is not suitable for production runs. Not only is the input simulated but also the APL system imposes several limitations. The necessity of employing several work-

spaces, which must be activated manually, greatly hinders the speed and utility of the overall system, as it requires a constant monitoring of the processing of a document line or page. For the PR system to be put into practical use several things would have to be done. (1) Build an electronic sensing device to be constructed and to operate after our paper model of the mechanical fovea. (2) Provide for the direct transmission to the computer or to a storage medium which the computer can use efficiently, e.g., magnetic tape, of the data resulting from the input device. (3) Reprogram the PR system in another language so that the storage of the routines and the manipulation of a large array will not be limited by the APL workspace memory size of 31,872 bytes. That is until such time as APL may either (a) provide larger workspaces or (b) provide for automatic activation and storage of workspaces within a function. About four to six times the workspace-size memory would be adequate, depending on the grid size, i.e., how many registrations the grid can generate. If another programming language could be used effectively so that an object code would be generated for the program, it would contribute to the efficiency of operation. Although having APL able to create an object code would contribute to speed, it is not so necessary as having it provide for continuity of

operation.

If the PR system was constructed as suggested above, we could imagine a document reading device which stores a page pattern matrix in the computer or on, say, magnetic tape. When processing begins, control is effected from a typewriter console where all the initializing instructions can be given at the outset of program operation. All the APL functions could be used as they stand by replacing the change of workspace (manually) instructions to direct transfer of control to the next routine. However, since we made little attempt at programming efficiency, it would be desirable to make program changes which add to efficiency without altering the essence of the operations before using the PR system for production work.

# CHAPTER V

# RESULTS AND CONCLUSIONS

## 5.1 Introduction

This chapter will describe the experiments made on the PR system and give the results. It will also present an evaluation of the PR system and our conclusions.

## 5.2 Tests and Results

A set of Capital letters and Arabic numerals were given to the PR system for the CHAR program to learn and subsequently to recognize. The capital letters were chosen from a standard print font as appears in Uhr's paper back edition of Pattern Recognition, 1966. The numerals were from the IBM pica 72 type ball. It will be noted that a print font tends to be "noisy" in the sense that extra strokes are added to characters which normally are not in free-hand printing. Furthermore various line widths are employed in a single character, whereas, say in a type font only one line width is used.

The letters and numerals were input into the computer using the hand-simulation of the mechanical fovea as described in Appendix A. Each letter was entered as a 25 × 40 pattern matrix and stored. Then the letters were learned. The results of learning, i.e., the category matrices, are presented in figures 13, 14, 15 and 16 in

a tabular form. The learning times ranged from 5.28 seconds to 1.33 seconds with an average of 3.18 seconds of IBM 360/67 processor time for the 37 patterns (the extra pattern is a |). (The time is for the normalize and define procedure, i.e., the function GO in workspace CHAR. If the complete PR system were timed it would have to include the manual transfer of control and exchange of data between workspaces.)

Before the patterns were recognized they were all entered into a 100 × 800 page matrix. Not all the characters were recognized correctly. It was thought necessary to remove two pattern representations from the category matrices C1 and C2, i.e., those for V and W because they appeared to conflict with Y and N respectively. However the representations for all numerals except 2 and 0 could be included with the letter representations in C1 and C2 without offering difficulty. After the two representations were removed, the CHAR program was capable of recognizing all the capital letters, except C, V, W, and Y and all the numerals except 2 and 0 with the alphanumeric specification. With the numeric specification the program could recognize all numerals.

The recognition times were nearly identical to the learning times as we should expect since the vector representation used for recognizing is determined by the

| PN | N1 | TL | BL | ML | LL | RL | CL | OB | OT | OR | OL | BCH | CHT | CHB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | A |  |  |  |  |  |  | 1 |  |  |  |  | 1 |  |
| 2 | B | 1 | 1 |  | 1 |  |  |  |  |  |  |  | 1 | 1 |
| 3 | C |  |  |  |  |  |  |  |  | 1 |  |  |  |  |
| 4 | D |  |  |  | 1 |  |  |  |  |  |  | 1 |  |  |
| 5 | E | 1 | 1 |  | 1 |  |  |  |  | 1 |  |  |  |  |
| 6 | F | 1 |  | 1 | 1 |  |  |  |  | 1 |  |  |  |  |
| 7 | G |  |  |  |  |  |  |  |  | 1 |  |  |  |  |
| 8 | H |  |  | 1 | 1 |  |  | 1 | 1 |  |  |  |  |  |
| 9 | I | 1 | 1 |  |  |  | 1 |  |  | 1 | 1 |  |  |  |
| 10 | J | 1 |  |  |  |  | 1 |  |  |  | 1 |  |  |  |
| 11 | K |  |  |  | 1 |  |  | 1 | 1 | 1 |  |  |  |  |
| 12 | L |  | 1 |  | 1 |  |  |  |  |  |  |  |  |  |
| 13 | M |  |  |  | 1 | 1 |  | 1 | 1 |  |  |  |  |  |
| 14 | N |  |  |  | 1 | 1 |  | 1 | 1 |  |  |  |  |  |
| 15 | O |  |  |  |  |  |  |  |  |  |  | 1 |  |  |
| 16 | P | 1 |  |  | 1 |  |  |  |  |  |  |  | 1 |  |
| 17 | Q |  | 1 |  |  |  |  |  |  | 1 |  |  | 1 |  |
| 18 | R | 1 |  |  | 1 |  |  | 1 |  | 1 |  |  | 1 |  |
| 19 | S |  |  |  |  |  |  |  |  | 1 | 1 |  |  |  |
| 20 | T | 1 |  |  |  |  | 1 |  |  |  |  |  |  |  |
| 21 | U |  |  |  | 1 | 1 |  |  | 1 |  |  |  |  |  |
| 22 | V |  |  |  |  |  |  |  | 1 |  |  |  |  |  |
| 23 | W |  |  |  |  |  |  | 1 | 1 |  |  |  |  |  |
| 24 | X |  |  |  |  |  |  | 1 | 1 | 1 | 1 |  |  |  |
| 25 | Y |  |  |  |  |  |  |  | 1 |  |  |  |  |  |
| 26 | Z | 1 | 1 |  |  |  |  |  |  | 1 | 1 |  |  |  |
| 27 | 1 |  | 1 |  |  |  | 1 |  |  | 1 | 1 |  |  |  |
| 28 | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 29 | 3 |  |  |  |  |  |  |  |  | 1 | 1 |  |  |  |
| 30 | 4 |  |  | 1 |  |  | 1 |  | 1 |  | 1 |  |  |  |
| 31 | 5 | 1 |  |  |  |  |  |  |  | 1 | 1 |  |  |  |
| 32 | 6 |  |  | 1 |  |  |  |  |  |  |  |  |  |  |
| 33 | 7 | 1 |  |  |  |  |  |  |  |  | 1 |  |  |  |
| 34 | 8 | 1 | 1 |  |  |  |  |  |  |  |  |  | 1 | 1 |
| 35 | 9 |  |  | 1 |  |  |  |  |  |  |  |  | 1 |  |
| 36 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 37 | \| |  |  |  |  |  | 1 |  |  |  |  |  |  |  |

Figure 13. The invariant category matrix, C1.

| PN | N1 | BOB | LOB | BOT | LOT | BOR | LORT | LORB | LORM | BOL | LOLT | LOLB | LOLM | XPT | IPT | DHB | DHT | DHR | DHL | TD | BD | ED | RD | LD | QD | WLC | WLR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | A | 1 | | | | | | | | | | | | | | 1 | | | | | | 1 | 1 | | | | |
| 2 | B | | | | | | | | 1 | | | | | | 1 | | | | | | | 1 | 1 | | | | |
| 3 | C | | | | | | | | 1 | | | | | | | | | 1 | | | 1 | | | 1 | | | |
| 4 | D | | | | | | | | | | | | | | | | | | | | | 1 | | 1 | | | |
| 5 | E | | | | | 1 | | | | | | | | | 1 | | | 1 | | 1 | | | | 1 | | | |
| 6 | F | | | | | | 1 | | | | | | | | | | | 1 | | 1 | | | | 1 | | | |
| 7 | G | | | | | | | | 1 | | | | | | 1 | | | 1 | | | 1 | | | 1 | | | |
| 8 | H | | 1 | | 1 | | | | | 1 | | | | | | 1 | 1 | | | 1 | | | 1 | | | | |
| 9 | I | | | | | 1 | | | | 1 | | | | | | | | 1 | 1 | | | 1 | | 1 | | | |
| 10 | J | | | | | | | | | 1 | | | | | | | | | 1 | 1 | | | | | 1 | | |
| 11 | K | | 1 | | 1 | 1 | | | | | | | | | | 1 | 1 | 1 | | | | 1 | | 1 | | | |
| 12 | L | | | | | | | | | 1 | | | | | | | | | | | 1 | | | 1 | | | |
| 13 | M | | 1 | | 1 | | | | | | | | | | | 1 | 1 | | | | | 1 | | | 1 | | |
| 14 | N | | 1 | | 1 | | | | | 1 | | | | | | 1 | 1 | | | 1 | | | | | 1 | | |
| 15 | O | | | | | | | | | | | | | | | | | | | | 1 | | | | 1 | | |
| 16 | P | | | | | | | | | 1 | | | | | | | | | | 1 | | | | 1 | | | |
| 17 | Q | | | | | | | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | |
| 18 | R | | 1 | | | | | | 1 | | | | | | | 1 | | 1 | | 1 | | | | 1 | | | |
| 19 | S | | | | | | | | 1 | 1 | | | | | 1 | | | 1 | 1 | | 1 | | 1 | | | | |
| 20 | T | | | | | | | | | | | | | | | | | | | 1 | | | | 1 | | | |
| 21 | U | | | | 1 | | | | 1 | | | | | | | | 1 | | | 1 | | | | 1 | | | |
| 22 | V | | | | 1 | | | | | | | | | | | | 1 | | | 1 | | | | 1 | | | |
| 23 | W | | 1 | | 1 | | | | | | | | | | | 1 | 1 | | | 1 | | | | | 1 | | |
| 24 | X | | 1 | | 1 | 1 | | | | 1 | | | | | 1 | 1 | 1 | 1 | 1 | | | 1 | | | 1 | | |
| 25 | Y | | | | 1 | | | | | | | | | | | | 1 | | | 1 | | | | 1 | | | |
| 26 | Z | | | | | 1 | | | | 1 | | | | | 1 | | | 1 | 1 | | 1 | | | | 1 | | |
| 27 | 1 | | | | | 1 | | | | 1 | | | | | | | | 1 | 1 | | | 1 | 1 | | | | |
| 28 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 29 | 3 | | | | | | | | 1 | 1 | | | | | 1 | | | 1 | 1 | 1 | | | 1 | | | | |
| 30 | 4 | | | | 1 | | | | 1 | | 1 | | | | | | 1 | | 1 | | | 1 | 1 | | | | |
| 31 | 5 | | | | | | 1 | | | | | 1 | | | 1 | | | 1 | 1 | 1 | | | | 1 | | | |
| 32 | 6 | | | | | | | | | | | | | | | | | | | | 1 | | | 1 | | | |
| 33 | 7 | | | | | | | | | 1 | | | | | | | | | 1 | 1 | | | | 1 | | | |
| 34 | 8 | | | | | | | | 1 | | | | 1 | | 1 | | | | | 1 | | | 1 | | | | |
| 35 | 9 | | | | | | | | | | | | | | | | | | | 1 | | | 1 | | | | |
| 36 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 37 | \| | | | | | | | | | | | | | | | | | | | | | 1 | | 1 | | 1 | 1 |

Figure 14. The not-so-invariant category matrix, C2.

| PN | N1 | TL | BL | ML | LL | RL | CL | OB | OT | OR | OL | BCH | CHT | CHB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | A | | | | | | | | | | | | | |
| 2 | B | | | | | | | | | | | | | |
| 3 | C | | | | | | | | | | | | | |
| 4 | D | | | | | | | | | | | | | |
| 5 | E | | | | | | | | | | | | | |
| 6 | F | | | | | | | | | | | | | |
| 7 | G | | | | | | | | | | | | | |
| 8 | H | | | | | | | | | | | | | |
| 9 | I | | | | | | | | | | | | | |
| 10 | J | | | | | | | | | | | | | |
| 11 | K | | | | | | | | | | | | | |
| 12 | L | | | | | | | | | | | | | |
| 13 | M | | | | | | | | | | | | | |
| 14 | N | | | | | | | | | | | | | |
| 15 | O | | | | | | | | | | | | | |
| 16 | P | | | | | | | | | | | | | |
| 17 | Q | | | | | | | | | | | | | |
| 18 | R | | | | | | | | | | | | | |
| 19 | S | | | | | | | | | | | | | |
| 20 | T | | | | | | | | | | | | | |
| 21 | U | | | | | | | | | | | | | |
| 22 | V | | | | | | | | | | | | | |
| 23 | W | | | | | | | | | | | | | |
| 24 | X | | | | | | | | | | | | | |
| 25 | Y | | | | | | | | | | | | | |
| 26 | Z | | | | | | | | | | | | | |
| 27 | 1 | | 1 | | | | 1 | | | 1 | 1 | | | |
| 28 | 2 | 1 | 1 | | | | | | | 1 | 1 | | | |
| 29 | 3 | | | | | | | | | 1 | 1 | | | |
| 30 | 4 | | | 1 | | | 1 | | 1 | | 1 | | | |
| 31 | 5 | 1 | | | | | | | | 1 | 1 | | | |
| 32 | 6 | | | 1 | | | | | | | | | | |
| 33 | 7 | 1 | | | | | | | | | 1 | | | |
| 34 | 8 | 1 | 1 | | | | | | | | | | 1 | 1 |
| 35 | 9 | | | 1 | | | | | | | | | 1 | |
| 36 | 0 | | | | | | | | | | | 1 | | |
| 37 | \| | | | | | | 1 | | | | | | | |

Figure 15. The invariant category matrix, C11.

| PN | N1 | BOB | LOB | BOT | LOT | BOR | LORT | LORB | LORM | BOL | LOLT | LOLB | LOLM | XPT | IPT | DHB | DHT | DHR | DHL | TD | BD | ED | RD | LD | QD | WLC | WLR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | A | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | B | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | C | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | D | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | E | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | F | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | G | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | H | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | I | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | J | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | K | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | L | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | M | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | N | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | O | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | P | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 17 | Q | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 18 | R | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 19 | S | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 | T | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 21 | U | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 22 | V | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 23 | W | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 24 | X | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 25 | Y | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 26 | Z | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 27 | 1 | | | | | 1 | | | | 1 | | | | | | | | 1 | 1 | | | 1 | 1 | | | | |
| 28 | 2 | | | | | 1 | | | | 1 | | | | | 1 | | | 1 | 1 | | 1 | | | | 1 | | |
| 29 | 3 | | | | | | | | 1 | 1 | | | | | 1 | | | 1 | 1 | 1 | | | 1 | | | | |
| 30 | 4 | | | | 1 | | | | 1 | | 1 | | | | | | 1 | | 1 | | | 1 | 1 | | | | |
| 31 | 5 | | | | | | 1 | | | | | 1 | | | 1 | | | 1 | 1 | 1 | | | | 1 | | | |
| 32 | 6 | | | | | | | | | | | | | | | | | | | | 1 | | | 1 | | | |
| 33 | 7 | | | | | | | | | 1 | | | | | | | | | 1 | 1 | | | | 1 | | | |
| 34 | 8 | | | | | | | | 1 | | | | 1 | | 1 | | | | | 1 | | | 1 | | | | |
| 35 | 9 | | | | | | | | | | | | | | | | | | | 1 | | | 1 | | | | |
| 36 | 0 | | | | | | | | | | | | | | | | | | | | 1 | | | | 1 | | |
| 37 | \| | | | | | | | | | | | | | | | | | | | | | 1 | | 1 | | 1 | 1 |

Figure 16. The not-so-invariant category matrix, C22.

same set of functions in the Define Routine, with possibly the exception of two functions, as are used in the learn mode. The recognize times ranged from 1.57 to 4.43 seconds with an average of 3.22 seconds for the 35 "acceptable" character patterns. Since the PR system was simulated in the APL language and APL runs in interpretive mode, these times may be higher than those used by a compiler language such as Fortran. Nevertheless the times compare favorably with other pattern recognition programs using the features approach. Grimsdale's program took 60 seconds on the Manchester Mark I computer; Uhr and Vossler's original program took 25 seconds for a comparison with five patterns and 40 seconds for a comparison with ten patterns on an IBM 709 computer. However in comparison with human performance our recognition times are decidedly slow. Recent investigations (Wilson, 1968) suggest that to equal human performance an approach using analysis by synthesis with context analysis should be employed.

The CHAR program has an accuracy of 94% recognition of the learned set of alphanumeric patterns after the representations for V and W have been deleted, i.e., thirty-one pattterns out of thirty-three were recognized correctly. However with the representations for V and W included at least six patterns would not be recognized in a learned set of thirty-five patterns for an accuracy

of 89%. For the numeric set of eleven patterns (| is included in the set) the accuracy is 100%. It is evident that the larger the learned set of patterns the more difficulty for attaining an accurate recognition.

## 5.3 Conclusions

From the results of our experiments the concepts used for the PR system appear to be basically sound. However it is evident that, as the size of the pattern class increases, difficulties may be encountered. These difficulties can be remedied by the addition of program modules which test for further critical features or by better utilization of the learned information. That is C, V, W and Y could be recognized by adding tests for width within a deep hole, for direction of slant, for two interior points from a horizontal aspect, and for solid centers (ones in the vicinity of the center). When a new feature is added it must be tested on a small set to insure reliable programming then that feature must be learned for the complete set of patterns. Most of the difficulties that were encountered with the CHAR program were due to the many possible cases which must be considered when constructing a features-test program. A better utilization of the learned information in the recognition routine could be effected in three ways:

(1) Use the complement of the vector representation, which

is determined for recognition purposes, with the complement of the category matrix (as the program is now using only half of the available information). (2) Use the results of the recognition methods, i.e., intersections of the results from intersections. (3) Further subdivide the features into three, instead of the present two, category matrices.

The PR system successfully extracts line matrices and pattern matrices for the CHAR program to operate upon. The definition parameters appear to be adequate; however, they give more reliable results when there is more information in the pattern matrix. In the few small pattern matrices that were tested, the resulting representation tends to contain extraneous features. The hexagonal element input grid gives good results when the pattern is adequately magnified but we cannot claim it to be superior as comparative tests were not made.

It is evident that many pattern sets of differing fonts should be used to test the CHAR program. However with a hand simulated input device this is not feasible. Considering the fact that fairly "noisy" patterns were used for our learning set this pattern recognition program appears to be quite successful. However further work should be done to improve the CHAR program and it is hoped that this will be done.

## BIBLIOGRAPHY

Adams, J.M., 1965. Optical Measurements in the Printing Industry, Pergamon Press Ltd., New York.

Arkadev, A.G. and E.M. Braverman, 1967. Computers and Pattern Recognition, translated from the Russian by W. Turski and J.D. Cowan, Thompson Book Company, Inc., Washington, D.C.

Barnhart, C.L., 1953. The American College Dictionary, Harper and Brothers Publishers, New York.

Bartley, H.S., 1963. Vision, Hafner Publishing Company, New York.

Bellman, R. and R. Kalaba, 1965. Dynamic Programming and Modern Control Theory, Academic Press, New York.

Benesh, R. and J. Benesh, 1956. An Introduction to Benesh Dance Notation, A. and C. Block, London.

Borko, H., 1962. Computer Applications in the Behavioral Sciences, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Buchsbaum, R., 1938. Animals without Backbones, The University of Chicago Press, Chicago.

Carne, E.B., 1965. Artificial Intelligence Techniques, Spartan Division (Books, Inc.), Washington, D.C.

Clowes, M.B. and J.R. Parks, 1961. "A New Technique in Automatic Character Recognition", The Computer Journal, 4:121-126.

Colwell, Robert N., 1965. "The Extraction of Data from Aerial Photographs by Human and Mechanical Means", Photogrammetria, 20:211-228.

Cox, A., 1964. A System of Optical Design, The Focal Press, London and New York.

Coxeter, H.S.M., 1961. Introduction to Geometry, John Wiley and Sons, Inc., New York.

Coxeter, H.S.M., 1963. Regular Polytopes, The Macmillan Company, New York.

Eden, M., 1962. "Handwriting and Pattern Recognition", IRE Transactions on Information Theory, IT-8:160-166.

Elsasser, W.M., 1958. The Physical Foundation of Biology, Pergamon Press, New York.

Falkoff, A.D. and K.E. Iverson, 1966. APL\360, International Business Machines Corp., Yorktown Heights, New York.

Feigenbaum, E.A. and J. Feldman, 1963. Computers and Thought, McGraw-Hill, Inc., New York.

Feinstein, A., 1958. Foundations of Information Theory, McGraw-Hill, Inc., New York.

Ferranti, Basil de, 1966. "The Human Brain", Computer Journal, 9:117-123.

Fischer, G.L., D.K. Pollock, B. Radack , and M.E. Stevens, 1962. Optical Character Recognition, Spartan Books, Washington, D.C.

George, F.H., 1962. The Brain as a Computer, Pergamon Press, Oxford.

Grimsdale, R.L. and J.M. Bullingham, 1961. "Character Recognition by Digital Computer using a Special Flying-Spot Scanner", The Computer Journal, 4:129-136.

Hartridge, H., 1950. Recent Advances in the Physiology of Vision, J. & A. Churchill Ltd., London.

Hebb, D.O., 1961. Organization of Behavior, John Wiley and Sons, Inc., New York.

Helmholtz, H. von, 1924. Physiological Optics, published by The Optical Society of America, I,II, & III.

Hubel, D.H. and T.N. Wiesel, 1965. "Receptive Fields and Functional Architecture in Two Non-Striate Visual Areas (18 and 19) of the Cat", J. Neurophysiol., 28:229-289.

Humphrey, G. and R.V. Coxon, 1963. The Chemistry of Thinking, Charles C. Thomas, publisher, Springfield, Illinois.

Hunt, E.B., J. Marin and P.J. Stone, 1966. Experiments in Induction, Academic Press, New York,

Hutchinson, Ann, 1954. Labanotation The System for Recording Movement, James Laughlin, publisher, New York.

Iverson, K.E., 1962. A Programming Language, John Wiley and Sons, Inc., New York.

Kabrisky, M., 1966. A Proposed Model for Visual Information Processing in the Human Brain, University of Illinois Press, Urbana and London.

Kingslake, R., 1965. Applied Optics and Optical Engineering, Academic Press, New York and London.

Kirsch, R.A., 1964. "Computer Interpretation of English Text and Picture Patterns", IEEE Transactions on Electronic Computers, EC-13:363-376.

Kullback, S., 1959. Information Theory and Statistics, John Wiley and Sons, Inc., New York.

Ledley, Robert S., 1966. "High-Speed Automatic Analysis of Biomedical Pictures", Science, 146.

Linksz, A., 1952. Physiology of the Eye, Vision, Grune and Stratton, New York, 2.

Longhurst, R.S., 1967. Geometrical and Physical Optics, Longmans, Green and Co. Ltd., London.

Mann, I. and A. Pirie, 1962. The Science of Seeing, Paterson Press, Perth, Western Australia.

McCulloch, W.S. and W.A. Pitts, 1943. "Logical Calculus of the Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics, 5:115-133.

Missotten, L., 1965. The Human Retina, Editions Arscia S.A., Brussels, Belgium.

Narasimhan, R., 1966. "Syntax-Directed Interpretation of Classes of Pictures", Communications of the ACM, 9.

Neisser, U., 1967. Cognitive Psychology, Appleton-Century-Crofts, New York.

Newman, E.A., 1961. "Some Comments on Character Recognition", The Computer Journal, 4:114-118.

Nilsson, Nils J., 1965. Learning Machines, McGraw-Hill Book Company, Toronto.

Osgood, C.E. and K.V. Wilson, 1961. Some Terms and Associated Measures for Talking About Human Communication, Institute of Communication Research, The University of Illinois, Urbana, Illinois.

Pedelty, M.J., 1963. An Approach to Machine Intelligence, Spartan Books, Washington, D.C.

Polyak, S., 1957. The Vertebrate Visual System, The Univeristy of Chicago Press, Chicago.

Ratliff, F., 1965. Mach Bands: Quantitative Studies on Neural Networks in the Retina, Holden-Day, Inc., San Francisco.

Rosenblatt, F., 1960. "Perceptron Simulation Experiments", Proc. of the IRE, 48:301-309.

Rubin, M.L. and G.L. Walls, 1965. Studies in Physiological Optics, Charles C. Thomas, publisher, Springfield, Illinois.

Sebestyen, G.S., 1962. Decision-Making Processes in Pattern Recognition, The Macmillan Company, New York.

Selfridge, O.G., 1955. "Pattern Recognition and Modern Computers", Proceedings of the 1955 Western Joint Computer Conference, 91-93.

Shannon, C.E. and J. McCarthy, 1956. Automata Studies, Princeton University Press, Princeton, New Jersey.

Tschermak-Seysenegg, A. Von, 1952. Introduction to Physiological Optics, translated by Paul Boeder, Charles C. Thomas, publisher, Springfield, Illinois.

Uhr, L., 1966. Pattern Recognition, John Wiley and Sons, Inc., New York.

Ullmann, J.R., 1964. "A Basic Approach to Pattern Recognition", Computer Journal, 7:282-289.

Vasicek, A., 1960. Optics of Thin Films, North-Holland Publishing Company, Amsterdam, Interscience Publishers, Inc., New York.

Vavilov, S.I., 1955. The Eye and the Sun, Foreign Language Publishing House, Moscow.

Wilson, K.V., 1968. "On the Importance of Grammars and Lists", paper presented at Pattern Recognition Conference, University of Manitoba, Winnipeg, May, 1968.

Wolken, J.J., 1966. Vision - Biophysics and Biochemistry of the Retinal Photoreceptors, Charles C. Thomas, publisher, Springfield, Illinois.

# APPENDIX A

## SIMULATION OF THE INPUT DEVICE

The pattern matrices used to test the CHAR program were the resultant configurations from a hand simulation of the input device, the mechanical fovea. The simulation was done using the paper model of the sensing grid as shown in figure A1. The characters were chosen from a font set and drawn to a scale approximating a magnification of twenty five times the original size. An example of a "magnified" character appears in figure A2. Then the hand simulation of the input device was performed as follows:

1) The magnified character drawing was placed under the paper mechanical fovea in the proper orientation.

2) The coder observed the h-elements where 10% or more of the element showed a dark spot from the pattern and recorded a one in the corresponding location on the coding sheet, where odd numbered row elements of the sensing grid give one entries in only the odd numbered columns on the coding sheet and even numbered row elements give entries in only the even numbered columns (see Section 3.2.4). All of the empty locations on the coding sheet are assumed zero. Figure A3 shows the figure of A2

superimposed on the sensing grid and the h-elements that are activated. Figure A4 is the recorded code from figure A3.

3) The binary data from the recorded code was entered into workspace STORE in the computer from and IBM 2741 typewriter remote terminal console. Figure A5 shows the pattern matrix as it is entered and stored and figure A6 shows the "normalized" pattern matrix which is operated upon by the CHAR program.

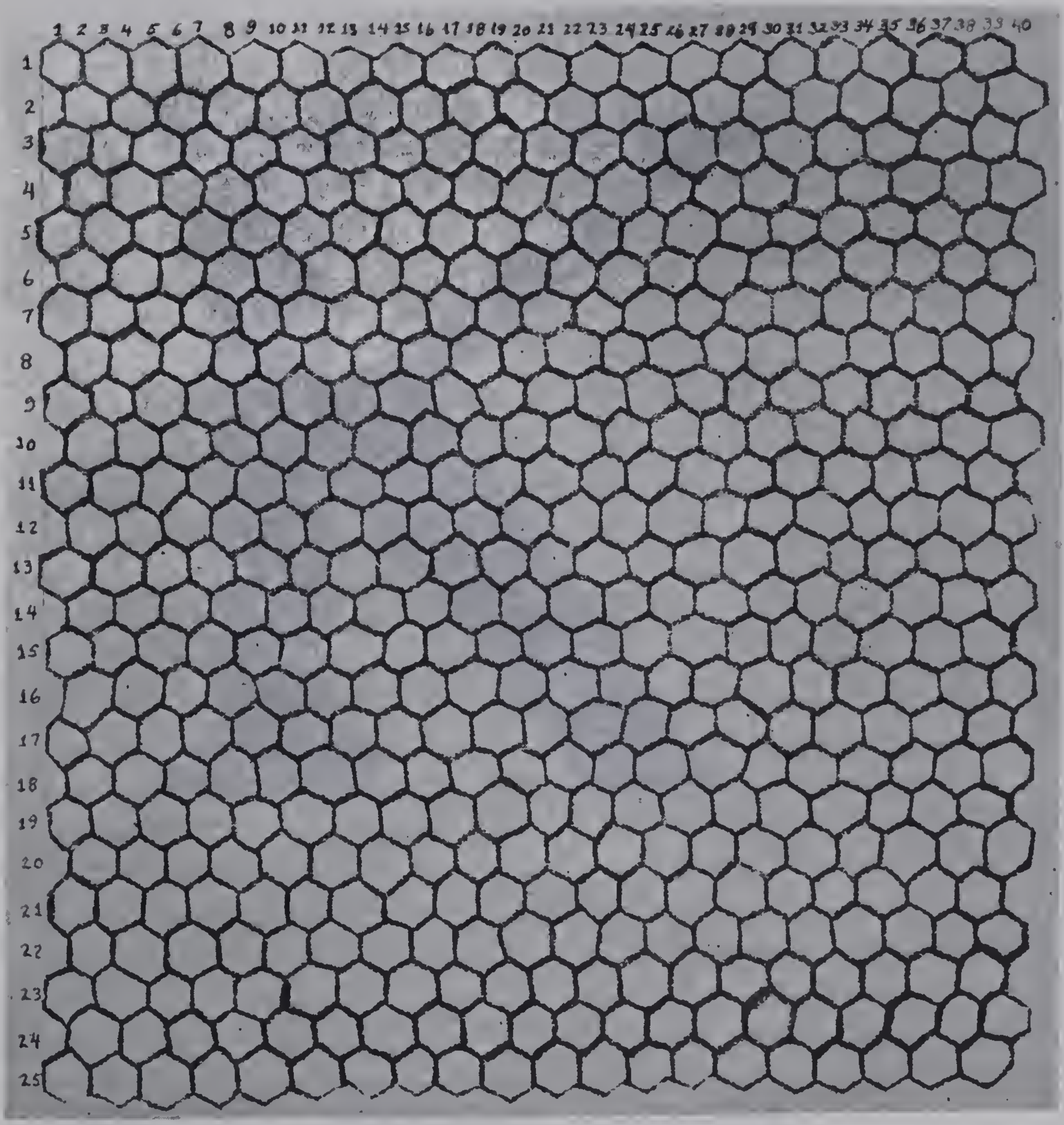


Figure A1. Mechanical fovea sensing grid.

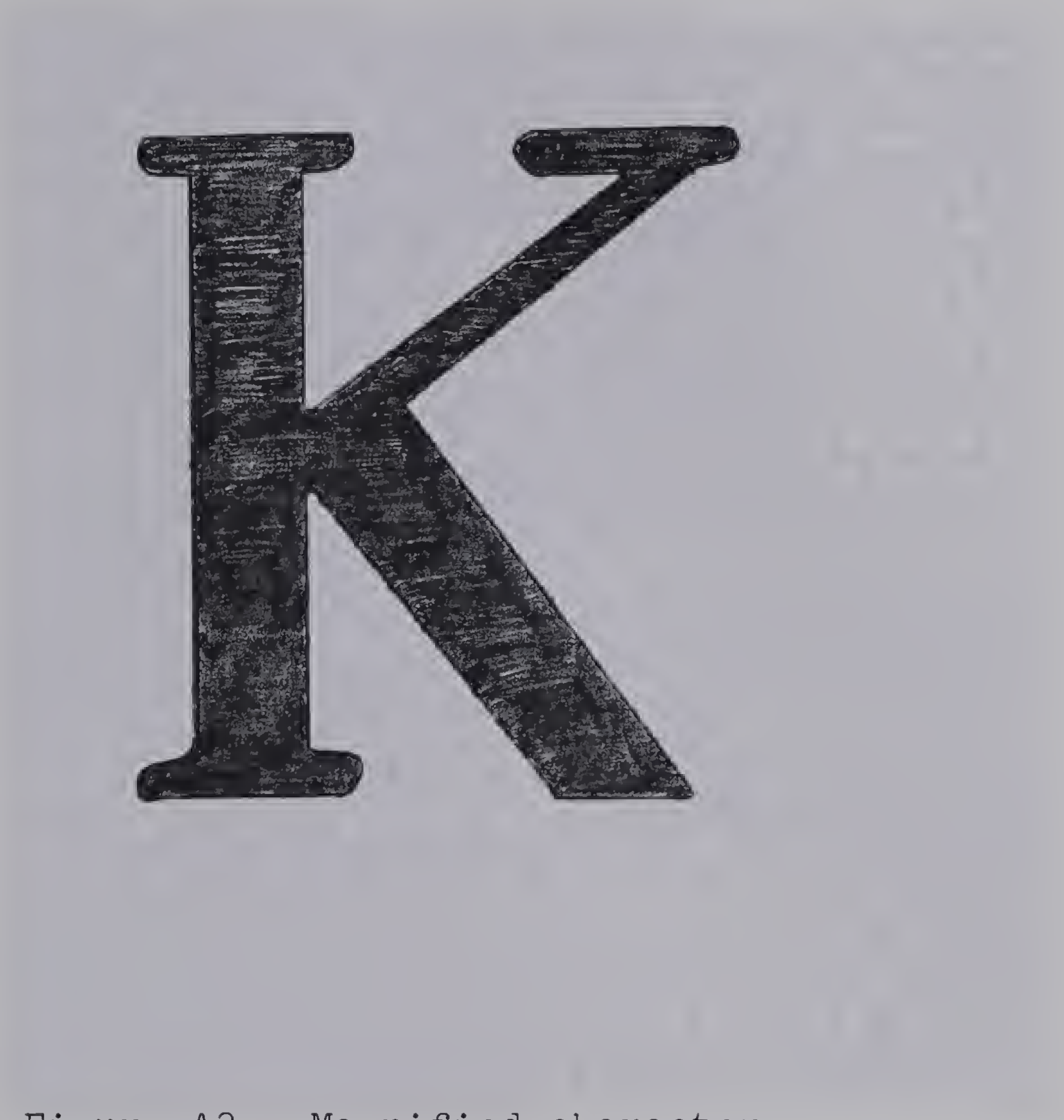


Figure A2. Magnified character.

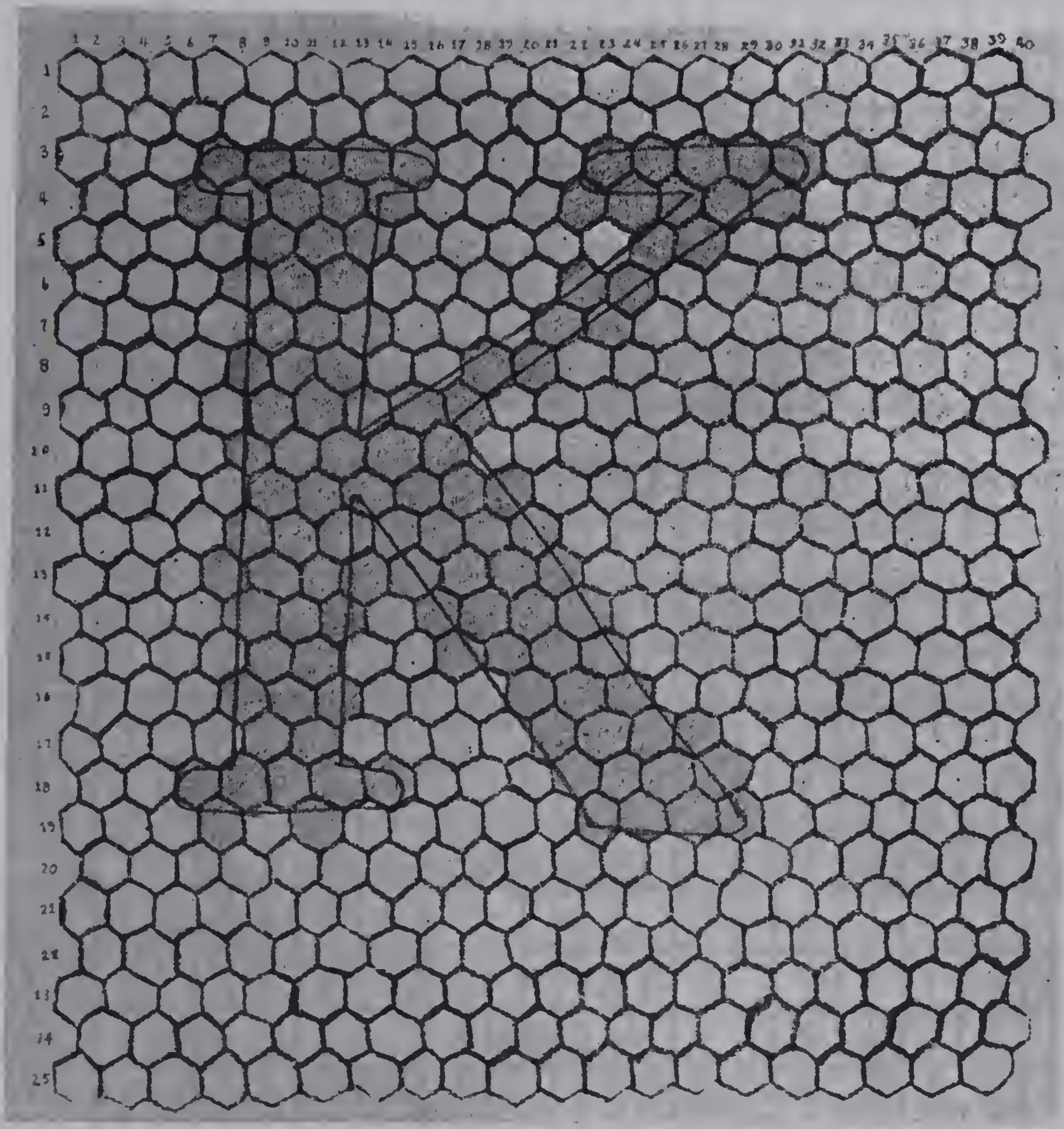


Figure A3. The sensing grid with the magnified character on it and the activated elements.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | |1| |1| |1| |1| |1| | | | | | | | |1| |1| |1| |1| |1| | | | | | | | | |
| 4 | | | | | |1| |1| |1| |1| |1| | | | | | | | |1| |1| |1| |1| |1| | | | | | | | | | |
| 5 | | | | | | | | |1| |1| |1| | | | | | | | | | | | |1| |1| | | | | | | | | | | | | |
| 6 | | | | | | | |1| |1| |1| | | | | | | | | | |1| |1| | | | | | | | | | | | | | | | |
| 7 | | | | | | | | |1| |1| |1| | | | | | | | |1| |1| | | | | | | | | | | | | | | | | |
| 8 | | | | | | | |1| |1| |1| | | | | |1| |1| | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | |1| |1| |1| |1| |1| | | | | | | | | | | | | | | | | | | | | | | |
| 10 | | | | | | | |1| |1| |1| |1| |1| | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | | | | | | | | |1| |1| |1| |1| |1| |1| | | | | | | | | | | | | | | | | | | | | | |
| 12 | | | | | | | |1| |1| |1| |1| |1| |1| |1| | | | | | | | | | | | | | | | | | | | | |
| 13 | | | | | | |1| |1| |1| |1| |1| |1| |1| |1| | | | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | |1| |1| |1| | | |1| |1| |1| |1| | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | |1| |1| | | | | |1| |1| |1| |1| | | | | | | | | | | | | | | | | |
| 16 | | | | | | | |1| |1| |1| | | | | | | |1| |1| |1| | | | | | | | | | | | | | | | |
| 17 | | | | | | | | |1| |1| | | | | | | | | |1| |1| |1| |1| | | | | | | | | | | | | |
| 18 | | | | | |1| |1| |1| |1| |1| | | | | | | | |1| |1| |1| |1| | | | | | | | | | | | |
| 19 | | | | | | |1| | | |1| | | | | | | | | | | |1| |1| |1| |1| | | | | | | | | | | |
| 20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 21 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 23 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 25 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Figure A4. Recorded code from the activated sensing grid.

*LIPGM*[1;13;;]

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure A5. A pattern matrix as it is stored in the computer.

*IPM*

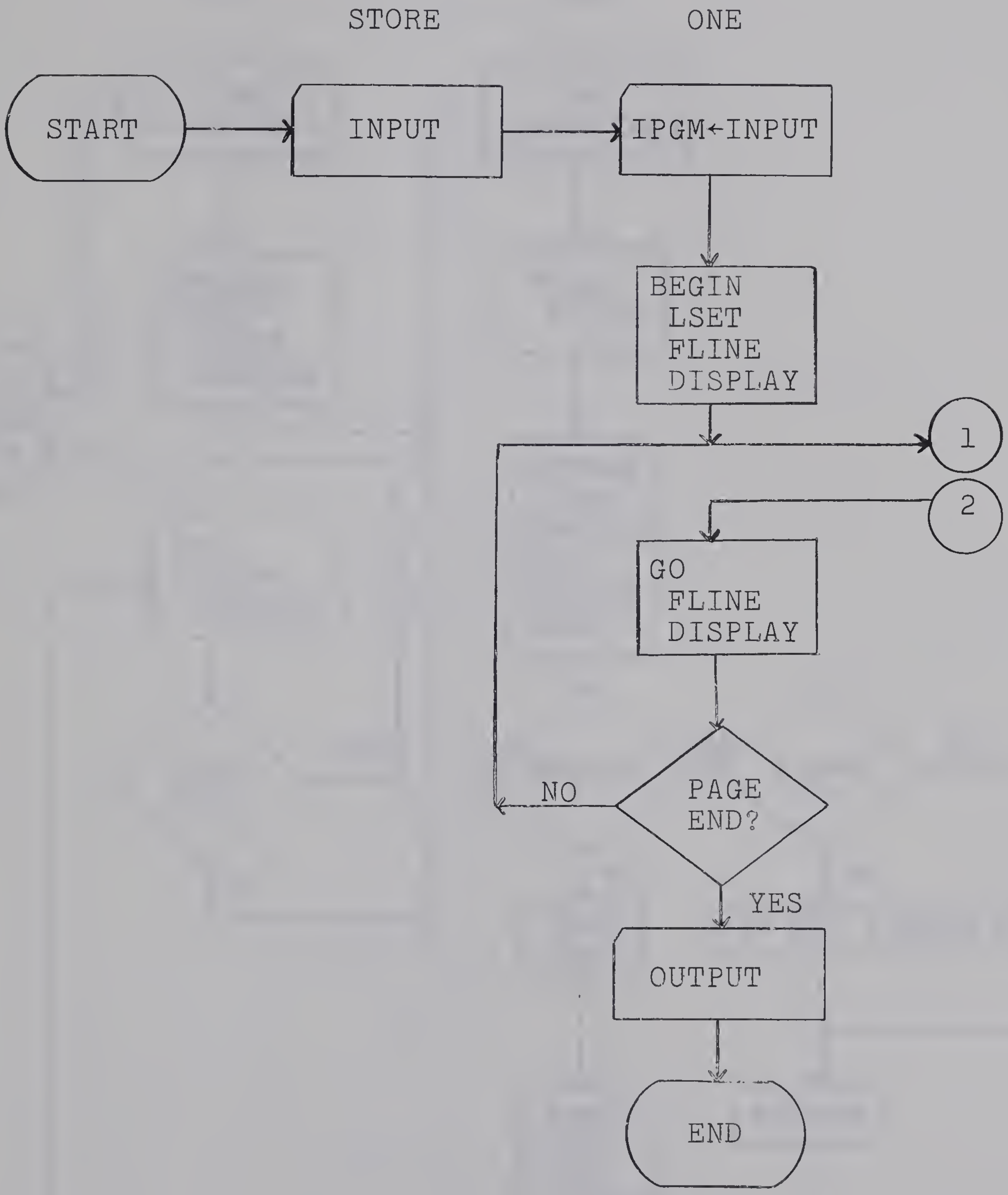
```
0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0
0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0
0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0
```

Figure A6. A normalized pattern matrix.

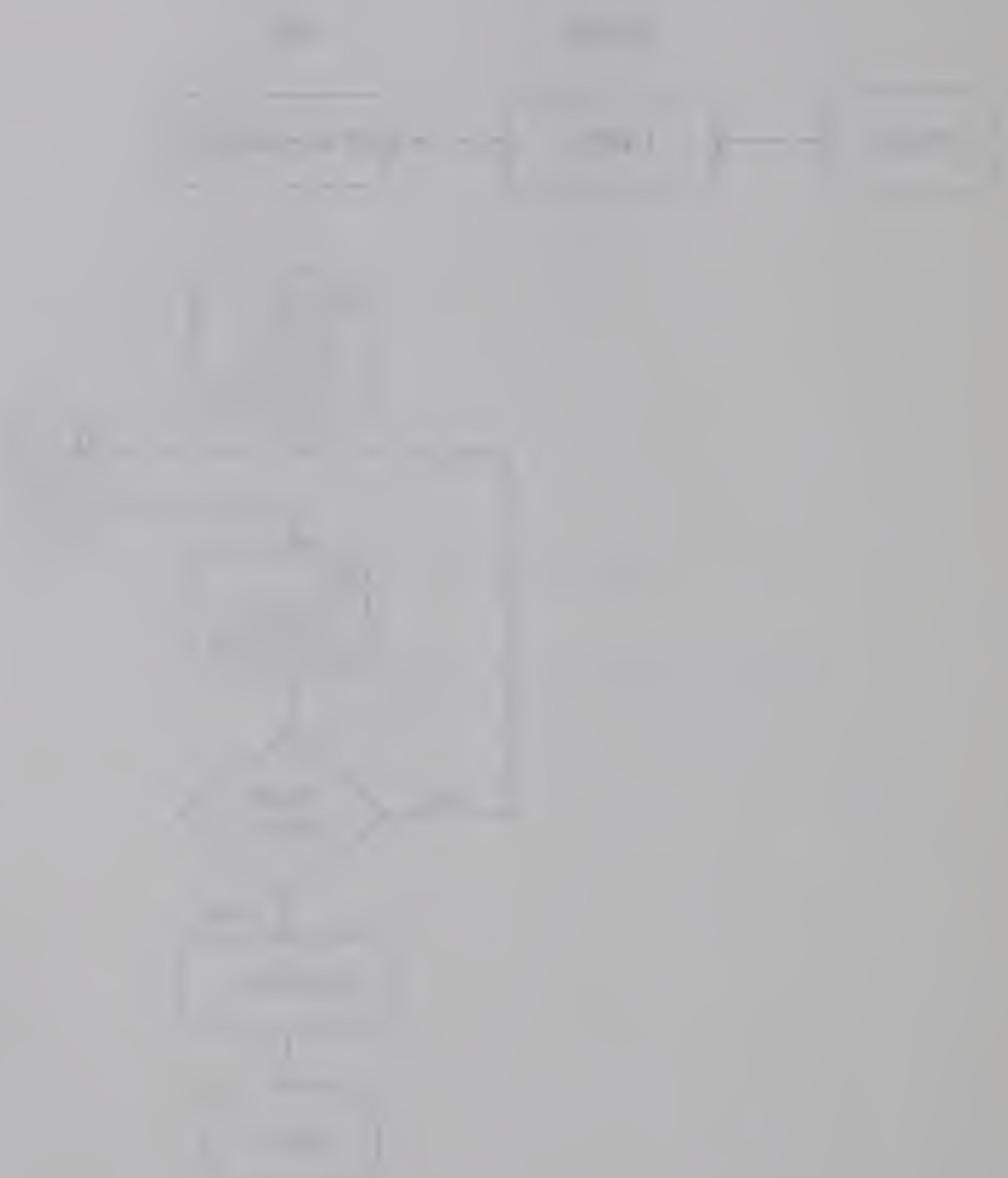
# APPENDIX B

## FLOW DIAGRAMS AND LISTINGS OF ROUTINES IN THE PR SYSTEM

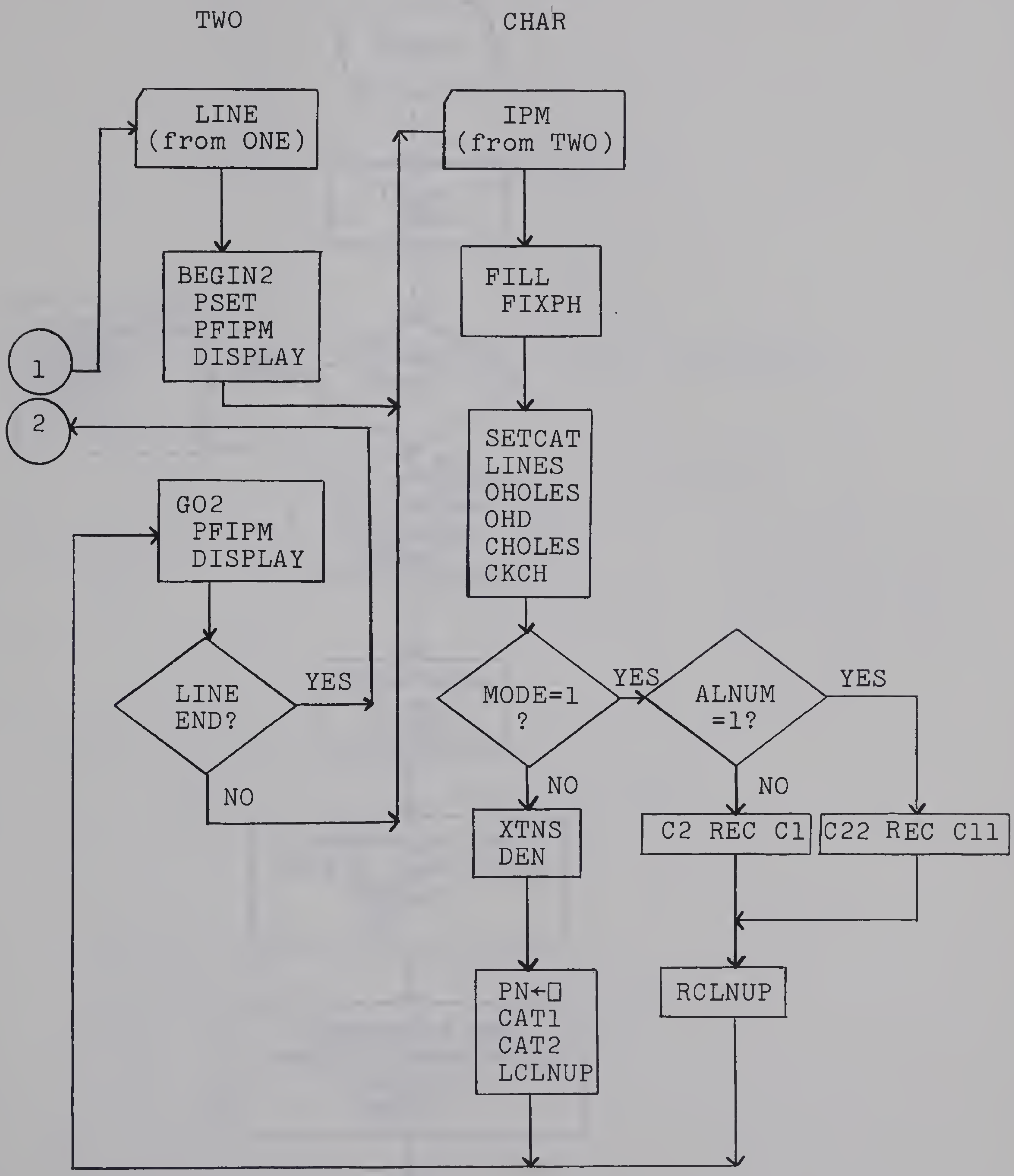
FLOW DIAGRAM OF THE PR SYSTEM



[illegible]

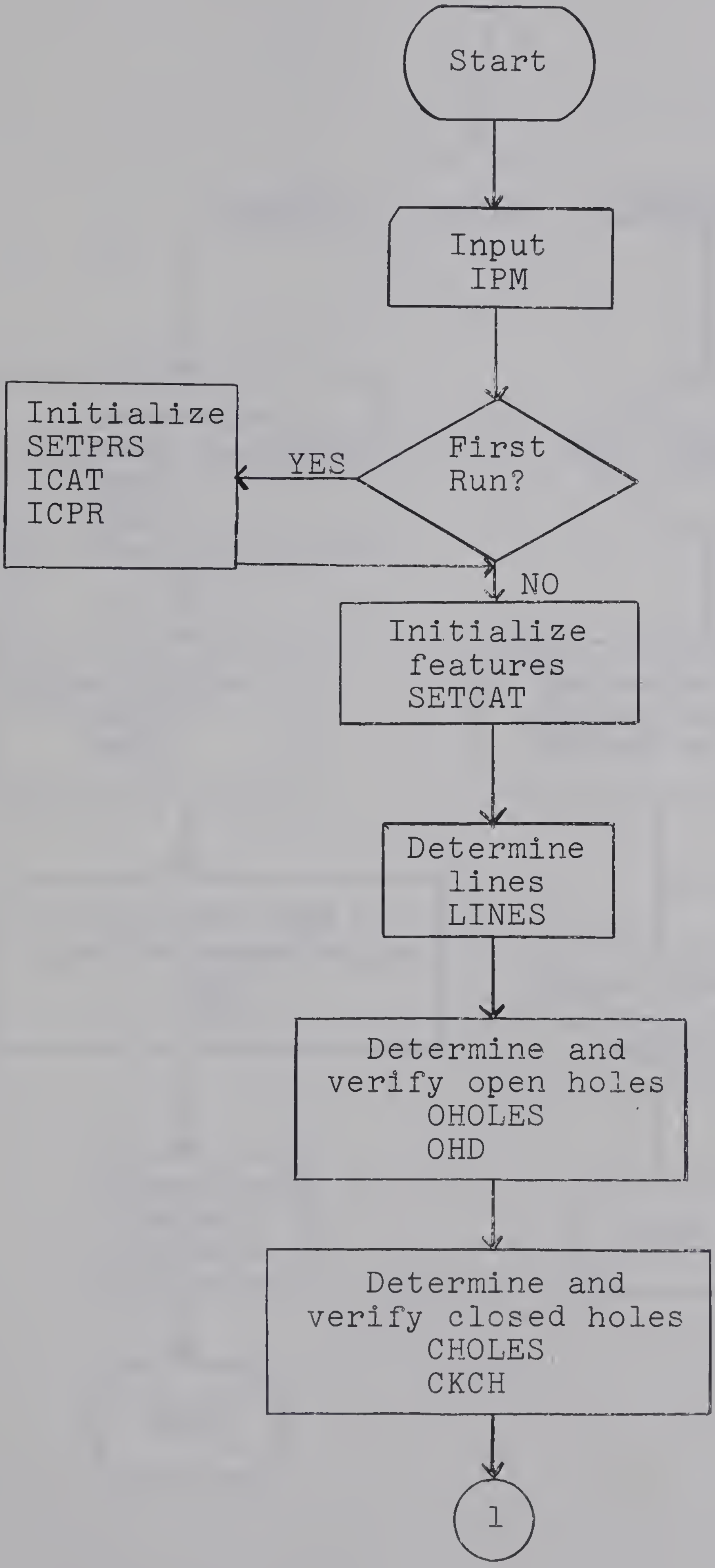


FLOW DIAGRAM OF THE PR SYSTEM (cont'd)



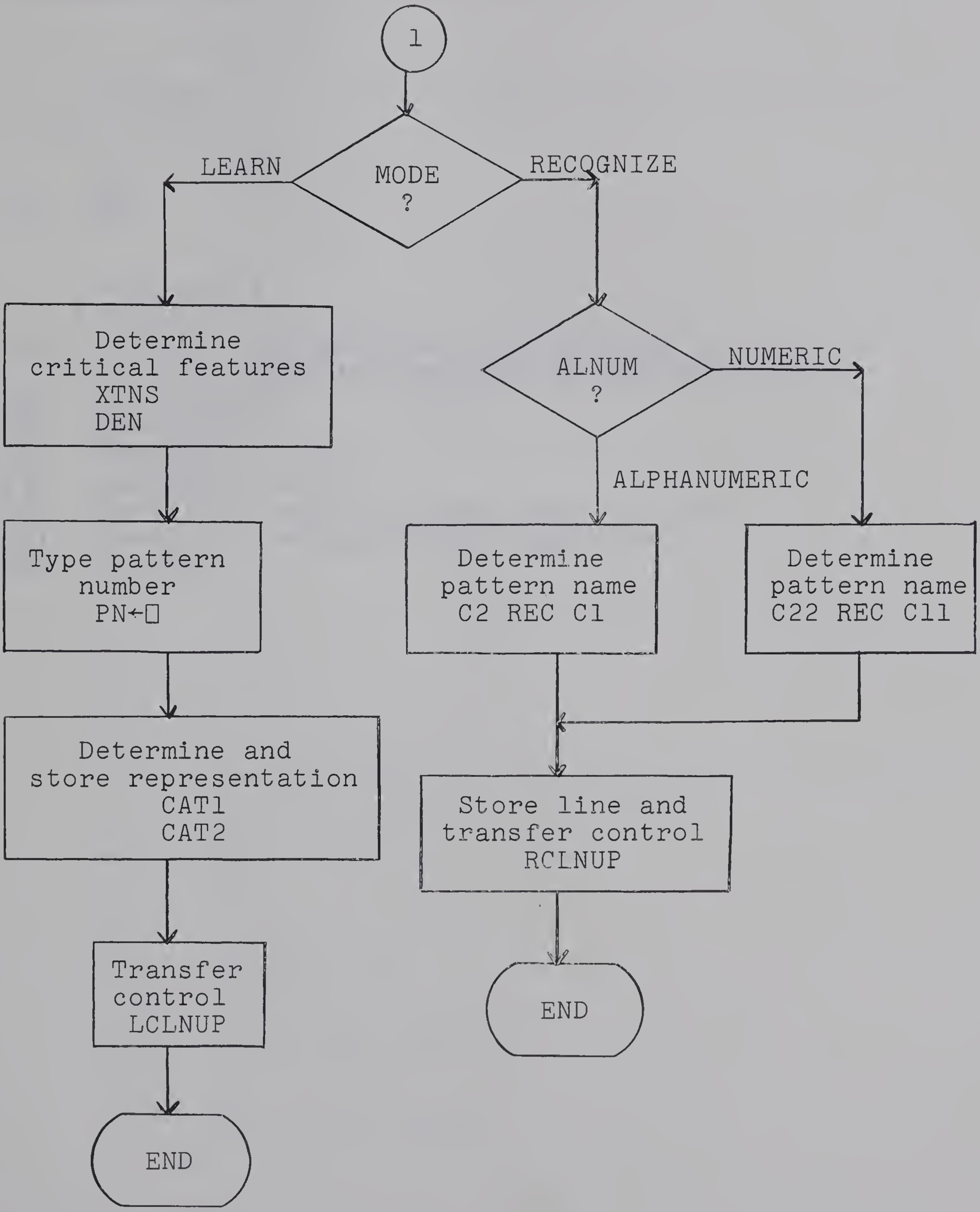
[illegible]

FLOW DIAGRAM OF THE CHAR PROGRAM


Start
Input IPM
First Run?
YES
NO
Initialize SETPRS ICAT ICPR
Initialize features SETCAT
Determine lines LINES
Determine and verify open holes OHOLES OHD
Determine and verify closed holes CHOLES CKCH
1

[illegible]

FLOW DIAGRAM OF THE CHAR PROGRAM (cont'd)


1
MODE ?
LEARN
RECOGNIZE
Determine critical features XTNS DEN
ALNUM ?
NUMERIC
ALPHANUMERIC
Type pattern number PN←⎕
Determine pattern name C2 REC C1
Determine pattern name C22 REC C11
Determine and store representation CAT1 CAT2
Store line and transfer control RCLNUP
Transfer control LCLNUP
END
END

[illegible]

## PROGRAM LISTINGS

Workspace STORE - stored pattern matrices.

```
      ρLIPGM
4  20  25  40

      ρM
100  800

      ∇PUTLM[⎕]∇
    ∇ R PUTLM C
[1]   'NOTE:  THIS FUNCTION WILL NOT WORK IF LIPGM AN
      D M ARE LARGE AND ARE BOTH IN THIS WORKSPACE'
[2]   S1←(ρSM)[1]
[3]   S2←(ρSM)[2]
[4]   PP←(ρM)[1]-R×S1
[5]   QQ←(ρM)[2]-C×S2
[6]   SM←((((R-1)×S1)ρ0),(S1ρ1),PPρ0)\[1]SM
[7]   SM←((((C-1)×S2)ρ0),(S2ρ1),QQρ0)\SM
[8]   M←M∨SM
    ∇
```

Workspace ONE - line extraction.

```
         ∇GETDATA[⎕]∇
       ∇ GETDATA
    [1]  'NOTE:  IPGM IS SHORT FOR INPUT PAGE MATRIX.  G
         ENERALLY THE IPGM NAME FOR LEARNING IS LIPGM AN
         D FOR
         RECOGNIZING IS M.'
    [2]  'INSTRUCTIONS:'
    [3]  '1.  REMOVE PREVIOUS IPGMS FROM WORKSPACE BY TY
         PING:∇LIPGM∇ AND ∇M∇.'
    [4]  '2.  INPUT DESIRED IPGM BY TYPING:  )COPY STORE
          LIPGM OR )COPY STORE M.'
    [5]  '3.  INITIALIZE THE OPERATIONAL IPGM BY TYPING:
           IPGM←M  OR  IPGM←LIPGM[I;J;;]   WHERE
         I IS THE ROW LOCATION AND J IS THE COLUMN LOCAT
         ION OF THE PATTERN.'
    [6]  '4.  TO START THE PROGRAM TYPE:  BEGIN'
       ∇


         ∇BEGIN[⎕]∇
       ∇ BEGIN
    [1]  'TYPE 1 OR 0;  1 FOR RECOGNIZE, 0 FOR LEARN'
    [2]  MODE←⎕
    [3]  'TYPE 1 OR 0; 1 FOR NUMERIC, 0 FOR ALPHANUMERIC'
    [4]  ALNUM←⎕
    [5]  SEE←0
    [6]  →(MODE=0)/9
    [7]  (MODE=1)/'TYPE 1 OR 0; 1 IF YOU WISH TO SEE INT
         ERMEDIATE RESULTS, 0 OTHERWISE'
    [8]  SEE←⎕
    [9]  NR←(ρIPGM)[1]
    [10] LSET
    [11] FLINE
    [12] →(PAGEND=1)/OUTPE
    [13] AS←ALNUM,SEE
    [14] DISPLAY
    [15] 'TYPE )COPY ONE MODE'
    [16] 'TYPE )COPY ONE AS'
    [17] 'TYPE BEGIN2'
    [18] →0
    [19] OUTPE:'EMPTY PAGE'
       ∇
```

Workspace ONE (Continued)

```
             ∇LSET[⎕]∇
     ∇ LSET
[1]    LC←0
[2]    LSD←0
[3]    LS←LSD≥+/IPGM
[4]    LS1←LS
[5]    LB←0
[6]    LE←0
[7]    LINE←(25 40)⍴0
     ∇


             ∇FLINE[⎕]∇
     ∇ FLINE
[1]    PAGEND←0
[2]    LOVERLAP←0
[3]    SLB←0
[4]    →(PAGEND←NR=+/LS)/0
[5]    LC←LC+1
[6]    LBS←SLB/LB
[7]    LB←LS⍳0
[8]    L0←LB
[9]    LB1←LB-1
[10]   LB2←NR-LB1
[11]   LE1←LS+(LB1⍴¯1),LB2⍴0
[12]   LE←(LE1⍳1)-1
[13]   LB←((SLB=0),SLB=1)/LB,LBS
[14]   LS1←((LB-1)⍴1),((LE-LB-1)⍴0),(NR-LE)⍴1
[15]   →(LOVERLAP←LE≥NR)/21
[16]   LS←(((⍴LS1)⍴SLB=1),(⍴LS)⍴SLB=0)/LS1,LS
[17]   LINE←((LE⍴~LS),(NR-LE)⍴0)/[1]IPGM
[18]   LS←LE1+(LE⍴1),(NR-LE)⍴0
[19]   →(SLB←(LE-LB)<5)/4
[20]   →0
[21]   LB←L0
[22]   LINE←(~LS)/[1]IPGM
     ∇
```

Workspace ONE (Continued)

```
     ∇DISPLAY[⎕]∇
     ∇ DISPLAY
[1]    TLBLE←LB,LE
[2]    'TYPE )SAVE ONE'
[3]    'TYPE )LOAD TWO'
[4]    'TYPE )COPY ONE LINE'
[5]    'TYPE )COPY ONE TLBLE'
     ∇


      ∇GO[⎕]∇
     ∇ GO
[1]    →(NR=+/LS)/ENDP
[2]    FLINE
[3]    DISPLAY
[4]    ' BEGIN2'
[5]    →0
[6]    ENDP:'PAGE FINISHED'
[7]    →(~MODE)/0
[8]    'TYPE: )COPY CHAR O
                  )COPY CHAR OUT
                  O
                  OUT'
     ∇
```

Workspace TWO - pattern extraction.

```
       ∇BEGIN2[⎕]∇
     ∇ BEGIN2
[1]    →(MODE=1)/4
[2]    'TYPE NAME OF PATTERN'
[3]    NAME←⍞
[4]    NC←(⍴LINE)[2]
[5]    PSET
[6]    PGROUPSP
[7]    PFIPM
[8]    DISPLAY2
[9]    'TYPE )COPY TWO MODE'
[10]   'TYPE )COPY TWO AS'
[11]   'TYPE GO'
     ∇

       ∇PSET[⎕]∇
     ∇ PSET
[1]    PC←0
[2]    PSD←0
[3]    PS←PSD≥+/[1]LINE
[4]    PB←0
[5]    PE←0
[6]    IPM←(25 40)⍴0
[7]    PSP←9
[8]    PSP1←0
[9]    PSPD←8
     ∇

       ∇PGROUPSP[⎕]∇
     ∇ PGROUPSP
[1]    →(PC=0)/4
[2]    PSP1←(PE-PB-1)+(PSPD÷PC)+PSP1
[3]    PSP←PSP1÷PC
[4]    PSW←PB+(PE-PB-1)+PSP
[5]    PGSP←0
[6]    PGB←PS⍳0
[7]    PGSP←PSW≤PGB-1
     ∇
```

Workspace TWO (Continued)

```
      ∇PFIPM[⎕]∇
    ∇ PFIPM
[1]   POVERLAP←0
[2]   PATTERNEND←0
[3]   →(PATTERNEND←NC=+/PS)/0
[4]   PC←PC+1
[5]   PB←PSι0
[6]   PB1←PB-1
[7]   PB2←NC-PB1
[8]   AGN:PE1←PS+(PB1ρ¯1),PB2ρ0
[9]   PE←(PE1ι1)-1
[10]  PEN←PE1+(PEρ1),(NC-PE)ρ0
[11]  →(POVERLAP←PE≥NC)/17
[12]  PBN←PENι0
[13]  PD←PBN-PE+1
[14]  →(PBN>NC)/16
[15]  →(PD≤2)/MORE
[16]  POVERLAP←((PD≤2),PD>2)/ 1 0
[17]  IPM←((PEρ~PS),(NC-PE)ρ0)/LINE
[18]  PS←PEN
[19]  →0
[20]  MORE:PS←PS+((PEρ0),(PDρ¯1),(NC-PBN-1)ρ0)
[21]  →AGN
    ∇


      ∇DISPLAY2[⎕]∇
    ∇ DISPLAY2
[1]   T←TLBLE,PB,PE,PGSP
[2]   'TYPE )SAVE TWO'
[3]   'TYPE )LOAD CHAR'
[4]   ⎕←(MODE=0)/'TYPE )COPY TWO NAME'
[5]   'TYPE )COPY TWO IPM'
[6]   'TYPE )COPY TWO T'
    ∇
```

Workspace TWO (Continued)

```
      ∇GO2[⎕]∇
    ∇ GO2
[1]   →(NC=+/PS)/GETL
[2]   →(MODE=1)/5
[3]   'TYPE NAME OF PATTERN'
[4]   NAME←⍞
[5]   PGROUPSP
[6]   PFIPM
[7]   DISPLAY2
[8]   'TYPE GO'
[9]   →0
[10]  GETL:'TYPE )SAVE TWO'
[11]  'TYPE )LOAD ONE'
[12]  'TYPE GO'
    ∇
```

Workspace CHAR - normalize, define, and learn or recognize.

Initialize

```
        ∇SETPRS[⎕]∇
      ∇ SETPRS
[1]     LPP←0.2
[2]     LPP1←1-LPP
[3]     LDP←0.77
[4]     LWMP←0.55
[5]     LHDP←0.12
[6]     BHDP←0.5
[7]     BDDP←0.25
[8]     LDDP←0.12
[9]     OHPP←1÷3
[10]    OHPPB←2÷3
[11]    MDP←0.5
      ∇
```

```
        ∇ICAT[⎕]∇
      ∇ ICAT
[1]     RR←4
[2]     CC←6
[3]     MP←(40,RR,CC)ρ0
[4]     C11←C1←(40 13)ρ0
[5]     C22←C2←(40 26)ρ0
[6]     N1←40ρ' '
[7]     O←(4 25)ρ' '
[8]     PN←0
[9]     LI←6ρ0
[10]    OUT←ι0
[11]    LZ←RZ←LCHT←LCHB←2ρ0
[12]    ICPR
      ∇
```

```
        ∇ICPR[⎕]∇
      ∇ ICPR
[1]     C←24ρ1
[2]     D←33ρ0
[3]     SS←40ρ0
      ∇
```

Workspace CHAR (Continued)

Mainline

```
        ∇GO[⎕]∇
      ∇ GO
[1]     ALNUM←AS[1]
[2]     SEE←AS[2]
[3]     PGS←T[5]
[4]     FILLIPM
[5]     SETCAT
[6]     LINES IPM
[7]     OHOLES IPM
[8]     0 OHD IPM
[9]     CHOLES IPM
[10]    CKCH
[11]    →(MODE=1)/21
[12]    XTNS IPM
[13]    DEN IPM
[14]    ('PN ';PN)
[15]    'TYPE PATTERN NUMBER'
[16]    PN←⎕
[17]    CAT1
[18]    CAT2
[19]    LCLNUP
[20]    →0
[21]    →(ALNUM=1)/25
[22]    C2 REC C1
[23]    RCLNUP
[24]    →0
[25]    C22 REC C11
[26]    →23
      ∇
```

Workspace CHAR (Continued)

Normalize

```
        ∇FILLIPM[⎕]∇
      ∇ FILLIPM
[1]     PW←T[4]-T[3]-1
[2]     PH←T[2]-T[1]-1
[3]     E←T[2]
[4]     B←T[1]
[5]     PHS←0=+/IPM
[6]     →((PHS[1]=1)∨PHS[PH]=1)/16
[7]     T←(1⌽IPM)∧¯1⌽IPM
[8]     T[;1,(⍴T)[2]]←0
[9]     PT←T∨IPM
[10]    PT←~PT∊0
[11]    T←(1⌽[1]PT)∧¯1⌽[1]PT
[12]    T[1,((⍴T)[1]);]←0
[13]    IPM←PT∨T
[14]    IPM←~IPM∊0
[15]    →0
[16]    FIXPH
[17]    PH←E-B-1
[18]    →7
      ∇


        ∇FIXPH[⎕]∇
      ∇ FIXPH
[1]     BO←0
[2]     SP←0
[3]     TT←IPM
[4]     →(PH=+/PHS)/16
[5]     BO←SP/B
[6]     B←PHS⍳0
[7]     T←PHS+((B-1)⍴¯1),(PH-B-1)⍴0
[8]     E←(T⍳1)-1
[9]     B←((SP=0),SP=1)/B,BO
[10]    S←((B-1)⍴1),((E-B-1)⍴0),(PH-E)⍴1
[11]    →(E≥PH)/18
[12]    PHS←(((⍴S)⍴SP=1),(⍴PHS)⍴SP=0)/S,PHS
[13]    TT←((E⍴~PHS),(PH-E)⍴0)/[1]IPM
[14]    PHS←T+(E⍴1),(PH-E)⍴0
[15]    →(SP←(E-B)<5)/4
[16]    IPM←TT
[17]    →0
[18]    IPM←(~S)/[1]IPM
[19]    PHS←S
      ∇
```

Workspace CHAR (Continued)
Define - initialize features and determine lines.

```
        ∇SETCAT[⎕]∇
      ∇ SETCAT
[1]     TL←BL←ML←RL←LL←CL←BOB←LOB←BOT←LOT←BOR←LOR←LORT←
        LORB←LORM←BOL←LOL←LOLT←LOLB←LOLM←0
[2]     BCH←CHT←CHB←WLC←WLR←XPT←IPT←DHB←DHT←DHR←DHL←TD←
        BD←ED←RD←LD←QD←0
[3]     LI←6ρ0
      ∇
```

```
        ∇LINES[⎕]∇
      ∇ LINES M
[1]     PVR←+/M
[2]     PVC←+/[1]M
[3]     INFO
[4]     →(DNS=0)/0
[5]     →(RD=0)/11
[6]     →(WLR←(MNR÷PH)≥LWMP)/11
[7]     M LIN(⌊/MPVR),PW,PH,0,(ρMRL),MRL,LPVR
[8]     TL←L1
[9]     BL←L2
[10]    ML←L3
[11]    →(CD=0)/17
[12]    →(WLC←(MNC÷PW)≥LWMP)/17
[13]    M LIN(⌊/MPVC),PH,PW,1,(ρMCL),MCL,LPVC
[14]    LL←L1
[15]    RL←L2
[16]    CL←L3
[17]    ML←((1=WLR∧WLC),0=WLR∧WLC)/(PH<PW),ML
[18]    CL←((1=WLR∧WLC),0=WLR∧WLC)/(PW≤PH),CL
      ∇
```

Workspace CHAR (Continued)
Define - determine lines (2).

```
        ∇INFO[⎕]∇
      ∇ INFO
 [1]    LPVC←LDP≤PVC÷PH
 [2]    LPVR←LDP≤PVR÷PW
 [3]    CD←∨/LPVC
 [4]    RD←∨/LPVR
 [5]    DNS←CD∨RD
 [6]    →(~DNS)/0
 [7]    MPVC←LPVC/PVC
 [8]    MPVR←LPVR/PVR
 [9]    MCL←LPVC/⍳PW
 [10]   MRL←LPVR/⍳PH
 [11]   MNR←+/LPVR
 [12]   MNC←+/LPVC
 [13]   CDATA
      ∇


        ∇CDATA[⎕]∇
      ∇ CDATA
 [1]    C[1]←C[1]+CD>LDP
 [2]    C[2]←C[2]+RD>LDP
 [3]    D[1]←D[1]+(RD>LDP)×RD
 [4]    D[2]←D[2]+(CD>LDP)×CD
 [5]    D[3]←D[3]+(LWMP≤MNC÷PW)×MNC÷PW
 [6]    D[4]←D[4]+(LWMP≤MNR÷PH)×MNR÷PH
 [7]    D[5]←D[5]+(CD>LDP)×(⌊/MCL)÷PW
 [8]    D[6]←D[6]+(CD>LDP)×(⌈/MCL)÷PW
 [9]    D[7]←D[7]+(RD>LDP)+(⌊/MRL)÷PW
 [10]   D[8]←D[8]+(RD>LDP)+(⌈/MRL)÷PW
      ∇
```

Workspace CHAR (Continued)
      Define - determine lines (3).

```
        ∇LIN[⎕]∇
      ∇ M LIN L
 [1]    TT←L[5]ρJ←((5ρ0),((ρL)-5)ρ1)/L
 [2]    J←((L[5]ρ0),((ρJ)-L[5])ρ1)/J
 [3]    L1←L2←L3←0
 [4]    →((L[1]÷L[2])<LDP)/0
 [5]    L1←((⌊/TT)÷L[3])≤LPP
 [6]    →(L1=0)/13
 [7]    L TLCY TT
 [8]    L1←V
 [9]    (ρJ)TLIN J
 [10]   →Q/0
 [11]   TT←J/ιρJ
 [12]   (ρJ)TLIN J
 [13]   L2←((⌈/TT)÷L[3])≥LPP1
 [14]   →(L2=0)/21
 [15]   L TLCY TT
 [16]   L2←V
 [17]   (ρJ)TLINφJ
 [18]   →Q/0
 [19]   TT←J/ιρJ←φJ
 [20]   →(0=ρTT)/0
 [21]   L3←1=∨/((TT÷L[3])<LPP1)∧(TT÷L[3])>LPP
 [22]   →(L3=0)/0
 [23]   →L[4]/30
 [24]   L[2]LG1(ρM)[2]ρM[(⌊/TT);]
 [25]   L3←PVR[⌊/TT]=E-B
 [26]   →L3/0
 [27]   TT←(I←~TT∊⌊/TT)/TT
 [28]   →(0=+/I)/0
 [29]   →21
 [30]   L[2]LG1(ρM)[1]ρM[;⌊/TT]
 [31]   L3←PVC[⌊/TT]=E-B
 [32]   →26
      ∇
```

Workspace CHAR (Continued)
Define - determine lines (4).

```
        ∇TLCY[⎕]∇
      ∇ L TLCY K
 [1]    →L[4]/9
 [2]    L[2]LG1(ρIPM)[2]ρIPM[(⌊/K);]
 [3]    V←PVR[⌊/K]=E-B
 [4]    →V/0
 [5]    K←(I←~K∊⌊/K)/K
 [6]    →(0=+I)/0
 [7]    →L[4]/9
 [8]    →2
 [9]    L[2]LG1(ρIPM)[1]ρIPM[;⌊/K]
 [10]   V←PVC[⌊/K]=E-B
 [11]   →4
      ∇


        ∇TLIN[⎕]∇
      ∇ K TLIN T
 [1]    K LG1 T
 [2]    SQ←E-B
 [3]    J←T+((B-1)ρ0),(SQρ¯1),(K-SQ+B-1)ρ0
 [4]    Q←SQ=(⌈/TT)-(⌊/TT)-1
      ∇


        ∇LG1[⎕]∇
      ∇ K LG1 T
 [1]    B←Tι1
 [2]    T←T+((B-1)ρ1),(K-B-1)ρ0
 [3]    E←Tι0
      ∇
```

Workspace CHAR (Continued)
      Define - determine open holes.

```
        ∇OHOLES[⎕]∇
      ∇ OHOLES M
[1]     I←0
[2]     I←I+1
[3]     K←PH+1-I
[4]     L←PW+1-I
[5]     →((BOB=1)∨LOB=1)/10
[6]     PW OH(ρM)[2]ρM[K;]
[7]     BOB←BO
[8]     LOB←LO
[9]     OBM←B+⌊(E-B)÷2
[10]    →((BOT=1)∨LOT=1)/15
[11]    PW OH(ρM)[2]ρM[I;]
[12]    BOT←BO
[13]    LOT←LO
[14]    OTM←B+⌊(E-B)÷2
[15]    →((BOR=1)∨LOR=1)/28
[16]    PH OH(ρM)[1]ρM[;L]
[17]    BOR←BO
[18]    LOR←LO
[19]    ORM←B+⌊(E-B)÷2
[20]    →((BOR=1)∨LOR=0)/28
[21]    OTBM L,ORM
[22]    LORT←OT
[23]    LORB←OB
[24]    LORM←OM
[25]    ORTM←OT1
[26]    ORBM←OT2
[27]    ORMM←OT3
[28]    →((BOL=1)∨LOL=1)/41
[29]    PH OH(ρM)[1]ρM[;I]
[30]    BOL←BO
[31]    LOL←LO
[32]    OLM←B+⌊(E-B)÷2
[33]    →((BOL=1)∨LOL=0)/41
[34]    OTBM I,OLM
[35]    LOLT←OT
[36]    LOLB←OB
[37]    LOLM←OM
[38]    OLTM←OT1
[39]    OLBM←OT2
[40]    OLMM←OT3
[41]    →(I≤⌈⌊/(PH,PW)÷4)/2
      ∇
```

Workspace CHAR (Continued)
      Define - determine open holes (2).

```
        ∇OH[⎕]∇
      ∇ K OH T
 [1]    K LG10 T
 [2]    E←((E>K),E≤K)/0,E
 [3]    BO←BHDP≤(E-B)÷K
 [4]    C[3]←C[3]+BO
 [5]    D[9]←D[9]+BO×(E-B)÷K
 [6]    LO←(BO=0)∧LHDP≤(E-B)÷K
 [7]    C[4]←C[4]+LO
 [8]    D[10]←D[10]+LO×(E-B)÷K
      ∇


        ∇LG10[⎕]∇
      ∇ K LG10 T
 [1]    B←T⍳1
 [2]    T←T+((B-1)⍴1),(K-B-1)⍴0
 [3]    B←T⍳0
 [4]    TT←T+((B-1)⍴¯1),(K-B-1)⍴0
 [5]    E←TT⍳1
      ∇
```

Workspace CHAR (Continued)
      Define - determine open holes (3).

```
      ∇OTBM[⎕]∇
    ∇ OTBM L
[1]   OT1←OT2←OT3←OT←OB←OM←TT←0
[2]   OT←OT∨OHPP>L[2]÷PH
[3]   OB←OB∨OHPPB<L[2]÷PH
[4]   OM←OM∨(OHPP≤L[2]÷PH)∧(L[2]÷PH)≤OHPPB
[5]   D[11]←D[11]+(OHPP>L[2]÷PH)×L[2]÷PH
[6]   C[5]←C[5]+OHPP>L[2]÷PH
[7]   D[12]←D[12]+(OHPPB<L[2]÷PH)×L[2]÷PH
[8]   C[6]←C[6]+OHPPB<L[2]÷PH
[9]   OT1←((OHPP>L[2]÷PH),OHPP≤L[2]÷PH)/(B+⌊(E-B)÷
      2),OT1
[10]  OT2←((OHPPB<L[2]÷PH),OHPPB≥L[2]÷PH)/(B+⌊(E-B)÷
      2),OT2
[11]  OT3←(((OHPP≤L[2]÷PH)∧(L[2]÷PH)≤OHPPB),(OHPP>L[
      2]÷PH)∨OHPPB<L[2]÷PH)/(B+⌊(E-B)÷2),OT3
[12]  →((PH=+/TT)∨E>PH-1)/0
[13]  PH LG10C(ρIPM)[1]ρIPM[;L[1]]
[14]  E←((E>PH),E≤PH)/0,E
[15]  →((E=0)∨B≥PH-1)/0
[16]  L[2]←B+⌊(E-B)÷2
[17]  PH LG1 TT
[18]  →2
    ∇


      ∇LG10C[⎕]∇
    ∇ K LG10C T
[1]   K LG10 T
[2]   K LG10 TT
[3]   E←((E>ρT),E≤ρT)/0,E
[4]   →(E=0)/7
[5]   TT←T+((E-1)ρ1),(K-E-1)ρ0
[6]   →0
[7]   TT←~TT
    ∇
```

Workspace CHAR (Continued)
Define - verify open holes.

```
       ∇OHD[⎕]∇
     ∇ T OHD M
[1]    →(0=BOB∨LOB∨BOT∨LOT∨BOR∨LOR∨BOL∨LOL)/0
[2]    →(0=BOB∨LOB)/13
[3]    B←((⌽(ρM)[1]ρM[;OBM])⍳1)÷PH
[4]    BOB←((B≥LDDP),B<LDDP)/BOB,0
[5]    C[9]←C[9]+BOB
[6]    D[15]←D[15]+B×BOB
[7]    LOB←((B≥LDDP),B<LDDP)/LOB,0
[8]    C[10]←C[10]+LOB
[9]    D[16]←D[16]+B×LOB
[10]   DHB←∨/BDDP≤B,(((⌽(ρM)[1]ρM[;OBM+2])⍳1),((⌽(ρM)[
       1]ρM[;OBM+4])⍳1),((⌽(ρM)[1]ρM[;OBM-2])⍳1),((⌽(ρ
       M)[1]ρM[;OBM-4])⍳1))÷PH
[11]   C[20]←C[20]+DHB
[12]   D[27]←D[27]+B×DHB
[13]   →(0=BOT∨LOT)/24
[14]   B←(((ρM)[1]ρM[;OTM])⍳1)÷PH
[15]   BOT←((B≥LDDP),B<LDDP)/BOT,0
[16]   C[24]←C[24]+BOT
[17]   D[17]←D[17]+B×BOT
[18]   LOT←((B≥LDDP),B<LDDP)/LOT,0
[19]   C[11]←C[11]+LOT
[20]   D[18]←D[18]+B×LOT
[21]   DHT←∨/BDDP≤B,((((ρM)[1]ρM[;OTM+2])⍳1),(((ρM)[1]
       ρM[;OTM+4])⍳1),(((ρM)[1]ρM[;OTM-2])⍳1),(((ρM)[1
       ]ρM[;OTM-4])⍳1))÷PH
[22]   C[21]←C[21]+DHT
[23]   D[28]←D[28]+B×DHT
[24]   →(0=BOR)/32
[25]   B←((⌽(ρM)[2]ρM[ORM;])⍳1)÷PW
[26]   BOR←((B≥LDDP),B<LDDP)/BOR,0
[27]   C[12]←C[12]+BOR
[28]   D[19]←D[19]+B×BOR
[29]   DHR←∨/BDDP≤B,(((⌽(ρM)[2]ρM[(ORM+2);])⍳1),((⌽(ρM
       )[2]ρM[(ORM+4);])⍳1),((⌽(ρM)[2]ρM[(ORM-
       2);])⍳1),((⌽(ρM)[2]ρM[(ORM-4);])⍳1))÷PW
[30]   C[22]←C[22]+DHR
[31]   D[29]←D[29]+B×DHR
[32]   →((BOR=1)∨LOR=0)/52
[33]   →(LORT=0)/39
[34]   B←((⌽(ρM)[2]ρM[ORTM;])⍳1)÷PW
[35]   LORT←((B≥LDDP),B<LDDP)/LORT,0
[36]   C[13]←C[13]+LORT
[37]   D[20]←D[20]+B×LORT
```

Workspace CHAR (Continued)
      Define - verify open holes (2).

```
[38]  DHR←B≥BDDP
[39]  →(LORB=0)/45
[40]  B←((⌽(ρM)[2]ρM[ORBM;])⍳1)÷PW
[41]  LORB←((B≥LDDP),B<LDDP)/LORB,0
[42]  C[14]←C[14]+LORB
[43]  D[21]←D[21]+B×LORB
[44]  DHR←DHR∨B≥BDDP
[45]  →(LORM=0)/51
[46]  B←((⌽(ρM)[2]ρM[ORMM;])⍳1)÷PW
[47]  LORM←((B≥LDDP),B<LDDP)/LORM,0
[48]  C[15]←C[15]+LORM
[49]  D[22]←D[22]+B×LORM
[50]  DHR←DHR∨B≥BDDP
[51]  LOR←0<LORT+LORB+LORM
[52]  →(0=BOL)/60
[53]  B←(((ρM)[2]ρM[OLM;])⍳1)÷PW
[54]  BOL←((B≥LDDP),B<LDDP)/BOL,0
[55]  C[16]←C[16]+BOL
[56]  D[23]←D[23]+B×BOL
[57]  DHL←∨/BDDP≤B,((((ρM)[2]ρM[(OLM+2);])⍳1),(((ρM)[
      2]ρM[(OLM+4);])⍳1),(((ρM)[2]ρM[(OLM-2);])⍳1),((
      (ρM)[2]ρM[(OLM-4);])⍳1))÷PW
[58]  C[23]←C[23]+DHL
[59]  D[30]←D[30]+B×DHL
[60]  →((BOL=1)∨LOL=0)/0
[61]  →(LOLT=0)/67
[62]  B←(((ρM)[2]ρM[OLTM;])⍳1)÷PW
[63]  LOLT←((B≥LDDP),B<LDDP)/LOLT,0
[64]  C[17]←C[17]+LOLT
[65]  D[24]←D[24]+B×LOLT
[66]  DHL←B≥BDDP
[67]  →(LOLB=0)/73
[68]  B←(((ρM)[2]ρM[OLBM;])⍳1)÷PW
[69]  LOLB←((B≥LDDP),B<LDDP)/LOLB,0
[70]  C[18]←C[18]+LOLB
[71]  D[25]←D[25]+B×LOLB
[72]  DHL←DHL∨B≥BDDP
[73]  →(LOLM=0)/79
[74]  B←(((ρM)[2]ρM[OLMM;])⍳1)÷PW
[75]  LOLM←((B≥LDDP),B<LDDP)/LOLM,0
[76]  C[19]←C[19]+LOLM
[77]  D[26]←D[26]+B×LOLM
[78]  DHL←DHL∨B≥BDDP
[79]  LOL←0<LOLT+LOLB+LOLM
```

Workspace CHAR (Continued)
Define - determine closed holes.

```
       ∇CHOLES[⎕]∇
     ∇ CHOLES M
[1]    →(DHL∨DHR)/6
[2]    V←(⌈PW÷2),(⌈PH÷2),BHDP
[3]    M CH V
[4]    →(BCH←VH∧HH)/0
[5]    LCHT←LCHB←2ρ0
[6]    →((BOT∨LOT)∧BOB∨LOB)/0
[7]    →(BOT∨LOT∨LORT∨LOLT)/30
[8]    J←1
[9]    J←J+1
[10]   →((J≥PH÷2)∧BOB∨LOB∨LORB∨LOLB)/0
[11]   →(J≥PH)/0
[12]   PW LG1 V←(ρM)[2]ρM[J;]
[13]   →(PVR[J]=E-B)/9
[14]   I←E
[15]   PW LG10 V
[16]   II←E
[17]   M CH I,J,LHDP
[18]   →(HH=0)/9
[19]   CHT←CHT∨(J<PH÷2)∧HH∧VH
[20]   CHB←CHB∨((J≥PH÷2)∧HH∧VH)∧~BOB∨LOB∨LORB∨LOLB
[21]   LCHT←((2ρCHT∧J<PH÷2),2ρ(~CHT)∨J≥PH÷2)/J,I,LCHT
[22]   LCHB←((2ρCHB∧J≥PH÷2),2ρ(~CHB)∨J<PH÷2)/J,I,LCHB
[23]   →(CHT∧CHB)/0
[24]   →CHB/0
[25]   →(CHT∧J<PH÷2)/29
[26]   I←I+1
[27]   →(I<II)/17
[28]   →9
[29]   →(BOB∨LOB∨LORB∨LOLB)/0
[30]   J←⌈PH÷2
[31]   →12
     ∇
```

Workspace CHAR (Continued)
Define - determine closed holes (2).

```
      ∇CH[⎕]∇
    ∇ M CH V
[1]   VH←0
[2]   PW LG10(ρM)[2]ρM[V[2];]
[3]   E←((E>PW),E≤PW)/0,E
[4]   HH←V[3]≤(E-B)÷PW
[5]   →(HH=0)/0
[6]   C[8]←C[8]+HH
[7]   D[14]←D[14]+HH×(E-B)÷PW
[8]   PH LG10(ρM)[1]ρM[;V[1]]
[9]   E←((E>PH),E≤PH)/0,E
[10]  VH←V[3]≤(E-B)÷PH
[11]  C[7]←C[7]+VH
[12]  D[13]←D[13]+VH×(E-B)÷PH
    ∇
```

Define - verify closed holes.

```
      ∇CKCH[⎕]∇
    ∇ CKCH
[1]   →(~(BCH∨CHT∨CHB))/0
[2]   →(~BCH)/5
[3]   (⌈PH÷2)CKC⌈PW÷2
[4]   BCH←T
[5]   →(~CHT)/8
[6]   LCHT[1]CKC LCHT[2]
[7]   CHT←T
[8]   →(~CHB)/0
[9]   LCHB[1]CKC LCHB[2]
[10]  CHB←T
    ∇
```

Workspace CHAR (Continued)
Define - verify closed holes (2).

```
       ∇CKC[⎕]∇
     ∇ I CKC J
[1]    R←4⍴0
[2]    PH LG10(⍴IPM)[1]⍴IPM[;J]
[3]    Q←E
[4]    (PW, 2 0 ,J)CK M←(⍴IPM)[2]⍴IPM[I;]
[5]    R[1]←L
[6]    (PW, 2 1 ,J)CK⌽M
[7]    R[3]←L
[8]    PW LG10(⍴IPM)[2]⍴IPM[I;]
[9]    Q←E
[10]   (PH, 1 0 ,I)CK M←(⍴IPM)[1]⍴IPM[;J]
[11]   R[2]←L
[12]   (PH, 1 1 ,I)CK⌽M
[13]   R[4]←L
[14]   T←∧/R
     ∇


       ∇CK[⎕]∇
     ∇ K CK I
[1]    K[1]LG1 I
[2]    →K[3]/7
[3]    L←∨/[K[2]]((K[4]⍴1),(K[1]-K[4])⍴0)/[K[
       2]]IPM
[4]    (⍴L)LG10 L
[5]    L←B≥Q
[6]    →0
[7]    L←∨/[K[2]](((K[1]-K[4])⍴1),K[4]⍴0)/[K[
       2]]⌽IPM
[8]    →4
     ∇
```

Workspace CHAR (Continued)
Define - determine points.

```
        ∇XTNS[⎕]∇
      ∇ XTNS M
[1]     →(0=BCH∨BOR∨LOR)/0
[2]     →(0=BCH)/6
[3]     PVC←+/[1]M
[4]     B←3>PVC[(ρPVC),((ρPVC)-1),(ρPVC)-2]
[5]     XPT←∨/B
[6]     IPM CIP PW,PH,(1÷3),(3÷4)
      ∇


        ∇CIP[⎕]∇
      ∇ M CIP V;T
[1]     I←⌈V[1]×V[3]
[2]     IPT←0
[3]     I←I+1
[4]     →(I>⌈V[1]×V[4])/0
[5]     →(M[1;I]=0)/3
[6]     B←M[;I]⍳0
[7]     T←M[;I]+((B-1)ρ¯1),(V[2]-B-1)ρ0
[8]     B←T⍳1
[9]     →(B≥V[2])/3
[10]    T←T+((B-1)ρ1),(V[2]-B-1)ρ0
[11]    B←T⍳0
[12]    →(B≥V[2])/3
[13]    T←T+((B-1)ρ¯1),(V[2]-B-1)ρ0
[14]    B←T⍳1
[15]    →(B>V[2])/3
[16]    IPT←1
[17]    →3
      ∇
```

Workspace CHAR (Continued)
      Define - determine density.

```
        ∇DEN[⎕]∇
      ∇ DEN M
[1]     E←⌊0.05×+/+/M
[2]     L←+/(⌈PW÷2)ρ+/[1]M
[3]     K←+/(⌈PW÷2)ρφ+/[1]M
[4]     T←+/(⌈PH÷2)ρ+/M
[5]     B←+/(⌈PH÷2)ρφ+/M
[6]     TD←E<T-B
[7]     BD←E<B-T
[8]     ED←E≥|B-T
[9]     RD←E<K-L
[10]    LD←E<L-K
[11]    QD←E≥|L-K
      ∇
```

Workspace CHAR (Continued)
      Define - learn.

```
        ∇CAT1[⎕]∇
      ∇ CAT1
[1]     OB←DHB∧BOB∨LOB
[2]     OT←DHT∧BOT∨LOT
[3]     OR←DHR∧BOR∨LOR
[4]     OL←DHL∧BOL∨LOL
[5]     VH←'TL BL ML LL RL CL OB OT OR OL BCH CHT CHB'
[6]     V←TL,BL,ML,LL,RL,CL,OB,OT,OR,OL,BCH,CHT,CHB
[7]     →(ALNUM=1)/11
[8]     C1[PN;]←V,((ρC1)[2]-(ρV))ρ0
[9]     N1[PN]←NAME
[10]    →0
[11]    C11[PN;]←V,((ρC11)[2]-(ρV))ρ0
[12]    →8
      ∇


        ∇CAT2[⎕]∇
      ∇ CAT2
[1]     VH2←'BOB LOB BOT LOT BOR LORT LORB LORM BOL LOL
        T LOLB LOLM XPT IPT DHB DHT DHR DHL TD BD ED RD
         LD QD WLC WL
        R'
[2]     V2←BOB,LOB,BOT,LOT,BOR,LORT,LORB,LORM,BOL,LOLT,
        LOLB,LOLM,XPT,IPT,DHB,DHT,DHR,DHL,TD,BD,ED,RD,
        LD,QD,WLC,WLR
[3]     →(ALNUM=1)/6
[4]     C2[PN;]←V2,((ρC2)[2]-(ρV2))ρ0
[5]     →0
[6]     C22[PN;]←V2,((ρC22)[2]-(ρV2))ρ0
[7]     →4
      ∇


        ∇LCLNUP[⎕]∇
      ∇ LCLNUP
[1]     'TYPE:
              )SAVE CHAR
              )LOAD TWO
              GO2'
      ∇
```

Workspace CHAR (Continued)
      Define - recognize.

```
      ∇REC[⎕]∇
    ∇ C2 REC C1
[1]   NC1←0
[2]   NC2←0
[3]   OL←DHL∧BOL∨LOL
[4]   OR←DHR∧BOR∨LOR
[5]   OT←DHT∧BOT∨LOT
[6]   OB←DHB∧BOB∨LOB
[7]   V←TL,BL,ML,LL,RL,CL,OB,OT,OR,OL,BCH,CHT,CHB
[8]   ANS←(∧/V/[2]C1)/N1
[9]   →(1≠ρANS)/13
[10]  S←PGS/' '
[11]  OUT←OUT,S,ANS
[12]  →0
[13]  NC1←1
[14]  XTNS IPM
[15]  DEN IPM
[16]  V2←BOB,LOB,BOT,LOT,BOR,LORT,LORB,LORM,BOL,LOLT,
      LOLB,LOLM,XPT,IPT,DHB,DHT,DHR,DHL,TD,BD,ED,RD,
      LD,QD,WLC,WLR
[17]  B←K←(∨/V2/[2]C2)/N1
[18]  NC2←((1≠ρK),1=ρK)/ 1 0
[19]  K←(ANS∊K)/ANS
[20]  →(1≠ρK)/23
[21]  ANS←((1≠ρK),(ρK)ρ1=ρK)/'?',K
[22]  →10
[23]  K←+/V2/C2
[24]  L←(K∊⌈/K)/N1
[25]  K←(ANS∊L)/ANS
[26]  →(1=ρK)/21
[27]  T←+/V/C1
[28]  ANS←(T∊⌈/T)/N1
[29]  →(1=ρANS)/10
[30]  K←(ANS∊L)/ANS
[31]  →(1=ρK)/21
[32]  K←(ANS∊B)/ANS
[33]  →(1=ρK)/21
[34]  K←B
[35]  →21
    ∇
```

Workspace CHAR (Continued)
      Define - recognize (2).

```
       ∇RCLNUP[⎕]∇
     ∇ RCLNUP
[1]    D[33]←D[33]+1
[2]    D[32]←D[32]+1
[3]    D[31]+D[31]+NC1
[4]    →(SEE=0)/7
[5]    ('PATTERN ';ANS)
[6]    ('LINE ';OUT)
[7]    →(25<ρOUT)/10
[8]    'TYPE:
              )SAVE CHAR
              )LOAD TWO
              GO2
       '
[9]    →0
[10]   'TYPE LINE NUMBER'
[11]   LN←⎕
[12]   O[LN;]←OUT
[13]   OUT←⍳0
[14]   →8
     ∇
```

## APPENDIX C

An example of the PR system learning a pattern.

```
      IPGM←LIPGM[1;3;;]
      BEGIN
TYPE 1 OR 0;  1 FOR RECOGNIZE, 0 FOR LEARN
⎕:
      0
TYPE 1 OR 0; 1 FOR NUMERIC, 0 FOR ALPHANUMERIC
⎕:
      0
TYPE )SAVE ONE
TYPE )LOAD TWO
TYPE )COPY ONE LINE
TYPE )COPY ONE TLBLE
TYPE )COPY ONE MODE
TYPE )COPY ONE AS
TYPE BEGIN2
      )SAVE ONE
ONE SAVED 21/04/68   16.42.42
      )LOAD TWO
TWO SAVED 21/04/68   16.39.05
      )COPY ONE LINE

      )COPY ONE TLBLE

      )COPY ONE MODE

      )COPY ONE AS

      BEGIN2
TYPE NAME OF PATTERN
A
TYPE )SAVE TWO
TYPE )LOAD CHAR
TYPE )COPY TWO NAME
TYPE )COPY TWO IPM
TYPE )COPY TWO T
TYPE )COPY TWO MODE
TYPE )COPY TWO AS
TYPE GO
      )SAVE TWO
TWO SAVED 21/04/68   16.43.53
```

An example of the PR system learning a pattern, (continued).

```
      )LOAD CHAR
CHAR SAVED 21/04/68   16.38.40
      )COPY TWO NAME

      )COPY TWO IPM

      )COPY TWO T

      )COPY TWO MODE

      )COPY TWO AS


      GO
PN 0
TYPE PATTERN NUMBER
⎕:
      1
TYPE:
      )SAVE CHAR
      )LOAD TWO
      GO2
      )SAVE CHAR
CHAR SAVED 21/04/68   16.46.37
      )LOAD TWO
TWO SAVED 21/04/68   16.43.53
      GO2
TYPE )SAVE TWO
TYPE )LOAD ONE
TYPE GO
      )SAVE TWO
TWO SAVED 21/04/68   16.47.03
      )LOAD ONE
ONE SAVED 21/04/68   16.42.42
      GO
PAGE FINISHED
```

An example of the PR system recognizing a series of letters and numerals.

```
      )COPY STORE M

      ρM
25  600
      IPGM←M
      BEGIN
TYPE 1 OR 0;  1 FOR RECOGNIZE, 0 FOR LEARN
⎕:
      1
TYPE 1 OR 0; 1 FOR NUMERIC, 0 FOR ALPHANUMERIC
⎕:
      0
TYPE 1 OR 0; 1 IF YOU WISH TO SEE INTERMEDIATE RESULTS
      , 0 OTHERWISE
⎕:
      1
TYPE )SAVE ONE
TYPE )LOAD TWO
TYPE )COPY ONE LINE
TYPE )COPY ONE TLBLE
TYPE )COPY ONE MODE
TYPE )COPY ONE AS
TYPE BEGIN2
      )SAVE ONE
ONE SAVED 21/04/68   16.25.03
      )LOAD TWO
TWO SAVED 21/04/68   16.17.29
      )COPY ONE LINE

      )COPY ONE TLBLE

      )COPY ONE MODE

      )COPY ONE AS
```

An example of the PR system recognizing a series of letters and numerals, (continued).

```
      BEGIN2
TYPE )SAVE TWO
TYPE )LOAD CHAR

TYPE )COPY TWO IPM
TYPE )COPY TWO T
TYPE )COPY TWO MODE
TYPE )COPY TWO AS
TYPE GO
      )SAVE TWO
TWO SAVED 21/04/68   16.28.11
      )LOAD CHAR
CHAR SAVED 21/04/68   16.20.34
      )COPY TWO IPM

      )COPY TWO T

      )COPY TWO MODE

      )COPY TWO AS



      GO
1
PATTERN A
LINE A
TYPE:
      )SAVE CHAR
      )LOAD TWO
      GO2

      )SAVE CHAR
CHAR SAVED 21/04/68   16.29.19
      )LOAD TWO
TWO SAVED 21/04/68   16.28.11
      GO2
TYPE )SAVE TWO
TYPE )LOAD CHAR

TYPE )COPY TWO IPM
TYPE )COPY TWO T
TYPE GO
```

An example of the PR system recognizing a series of letters and numerals, (continued).

```
      )SAVE TWO
TWO SAVED 21/04/68   16.31.04
      )LOAD CHAR
CHAR SAVED 21/04/68   16.29.19
      )COPY TWO IPM

      )COPY TWO T

      I21
2249
      GO
1
PATTERN P
LINE AP
TYPE:
      )SAVE CHAR
      )LOAD TWO
      GO2

      I21
2477
      )SAVE CHAR
CHAR SAVED 21/04/68   16.32.12
      )LOAD TWO
TWO SAVED 21/04/68   16.31.04
      GO2
TYPE )SAVE TWO
TYPE )LOAD CHAR

TYPE )COPY TWO IPM
TYPE )COPY TWO T
TYPE GO
      )SAVE TWO
TWO SAVED 21/04/68   16.32.40
      )LOAD CHAR
CHAR SAVED 21/04/68   16.32.12
      )COPY TWO IPM

      )COPY TWO T
```

An example of the PR system recognizing a series of letters and numerals, (continued).

```
      I21
2502
      GO
0
PATTERN R
LINE APR
TYPE:
      )SAVE CHAR
      )LOAD TWO
      GO2

      I21
2693
      )SAVE CHAR
CHAR SAVED 21/04/68   16.35.15
      )LOAD TWO
TWO SAVED 21/04/68   16.32.40
      GO2
TYPE )SAVE TWO
TYPE )LOAD CHAR

TYPE )COPY TWO IPM
TYPE )COPY TWO T
TYPE GO
      )SAVE TWO
TWO SAVED 21/04/68   16.35.42
      )LOAD CHAR
CHAR SAVED 21/04/68   16.35.15
      )COPY TWO IPM

      )COPY TWO T

      I21
2718
      GO
1
PATTERN 6
LINE APR 6
TYPE:
      )SAVE CHAR
      )LOAD TWO
      GO2

      I21
2941
```

An example of the PR system recognizing a series of letters and numerals, (continued).

```
      )SAVE CHAR
CHAR SAVED 21/04/68   16.37.16
      )LOAD TWO
TWO SAVED 21/04/68   16.35.42
      GO2
TYPE )SAVE TWO
TYPE )LOAD CHAR

TYPE )COPY TWO IPM
TYPE )COPY TWO T
TYPE GO
      )SAVE TWO
TWO SAVED 21/04/68   16.37.44
      )LOAD CHAR
CHAR SAVED 21/04/68   16.37.16
      )COPY TWO IPM

      )COPY TWO T

      ⌶21
2966
      GO
1
PATTERN 8
LINE APR 68
TYPE:
      )SAVE CHAR
      )LOAD TWO
      GO2

      ⌶21
3227
      )SAVE CHAR
CHAR SAVED 21/04/68   16.38.40
      )LOAD TWO
TWO SAVED 21/04/68   16.37.44
      GO2
TYPE )SAVE TWO
TYPE )LOAD ONE
TYPE GO
      )SAVE TWO
TWO SAVED 21/04/68   16.39.05
      )LOAD ONE
ONE SAVED 21/04/68   16.25.03
```

An example of the PR system recognizing a series of letters and numerals, (continued).

```
       GO
PAGE FINISHED
TYPE: )COPY CHAR O
               )COPY CHAR OUT
               O
               OUT
       )COPY CHAR O

       )COPY CHAR OUT

       O





       OUT
APR 68
```